PENENTUAN WAKTU TERAKHIR PENGGUNAAN GANJA DENGAN METODE *RADIAL BASIS FUNCTION NEURAL NETWORK* (RBFNN)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Sukma Fardhia Anggraini
NIM: 155150200111177

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

PENENTUAN WAKTU TERAKHIR PENGGUNAAN GANJA DENGAN METODE *RADIAL BASIS FUNCTION NEURAL NETWORK* (RBFNN)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Sukma Fardhia Anggraini
NIM: 155150200111177

Skripsi ini telah diuji dan dinyatakan lulus pada
27 Desember 2018
Telah diperiksa dan disetujui oleh:

Pembimbing I	Pembimbing II
Sigit Adinugroho S.Kom., M.Sc.	Randy Cahya Wihandika, S. ST., M.Kom.
NIK: 2016078807011001	NIK: 201405 880206 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D.
NIP: 197105182003121001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar referensi.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 27 Desember 2018

Sukma Fardhia Anggraini

NIM: 155150200111177

PRAKATA

Puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat, taufik dan hidayah-Nya sehingga laporan skripsi yang berjudul "Penentuan Waktu Terakhir Penggunaan Ganja dengan Metode *Radial Basis Function Neural Network* (RBFNN)" ini dapat terselesaikan.

Penulis menyadari bahwa skripsi ini tidak akan berhasil tanpa bantuan dan peran serta beberapa pihak. Oleh karena itu, penulis ingin menyampaikan rasa hormat dan terima kasih kepada:

1. Bapak Sigit Adinugroho, S.Kom., M.Sc. dan Bapak Randy Cahya Wihandika, S.ST., M.Kom. selaku Pembimbing skripsi yang telah dengan sabar dan pengertian membimbing serta mengarahkan penulis sehingga dapat menyelesaikan skripsi ini,
2. Bapak Agus Wahyu Widodo, S.T., M.Cs. selaku ketua Program Studi Teknik Informatika,
3. Bapak Tri Astoto Kurniawan, S.T., M.T., Ph.D. selaku ketua Jurusan Teknik Informatika,
4. Bapak Bayu Priyambadha, S.Kom., M.Kom., selaku dosen Penasihat Akademik yang selalu memberikan nasihat kepada penulis selama menempuh masa studi,
5. Ayah dan Ibu dari penulis atas segala dukungan, nasihat, perhatian dan kesabarannya dalam membesarkan dan mendidik penulis, serta yang senantiasa tiada hentinya memberikan doa dan semangat dalam mengerjakan laporan skripsi ini,
6. Seluruh civitas akademika Teknik Informatika Universitas Brawijaya yang telah banyak memberi bantuan dan dukungan selama penyelesaian laporan skripsi ini,
7. Dheby Tata Artha yang selalu memberikan dukungan moril, semangat dan doa untuk penulis,
8. Teman-teman mahasiswa Teknik Informatika 2015 yang selalu memberi semangat dan mendukung penulis untuk hal kebaikan.

Penulis menyadari bahwa dalam penyusunan laporan ini masih banyak kekurangan, sehingga saran dan kritik yang membangun sangat penulis harapkan. Akhir kata penulis berharap skripsi ini dapat membawa manfaat bagi semua pihak yang mebutuhkannya.

Malang, 27 Desember 2018

Penulis

Email: sukmafardhia@gmail.com

ABSTRAK

Sukma Fardhia Anggraini, Penentuan Waktu Terakhir Penggunaan Ganja dengan Metode *Radial Basis Function Neural Network* (RBFNN)

Pembimbing: Sigit Adinugroho, S.Kom, M. Sc dan Randy Cahya Wihandika, S.ST, M.Kom.

Pada Pada tahun 2017, jumlah penyalahguna ganja di Indonesia mencapai 1.742.285 orang. Ketika seorang pecandu ganja ingin menghentikan pemakaian secara mendadak, maka dapat menimbulkan gejala sakau. Untuk mengantisipasi gejala sakau maka dapat dilakukan langkah rehabilitasi, sehingga pecandu bisa mendapatkan perawatan yang komprehensif. Penentuan jenis rehabilitasi yang sesuai dapat menjadikan perawatan lebih efektif. Sehingga dengan mengetahui waktu terakhir penggunaan ganja, diharapkan dapat memberi informasi pendukung untuk menentukan program rehabilitasi yang sesuai untuk pecandu ganja. Salah satu teknik dalam *data mining* yang dapat digunakan untuk menyelesaikan permasalahan ini adalah dengan teknik klasifikasi. Pada penelitian ini, metode klasifikasi yang digunakan adalah metode *Radial Basis Function Neural Network* (RBFNN) dengan *K-Means*. Tahapan yang dilakukan meliputi tahap normalisasi data, metode *K-means* untuk menentukan nilai *center* dan *spread* pada fungsi aktivasi *Gaussian*, tahap *training* RBFNN dan tahap *testing* RBFNN. Penelitian ini menggunakan 627 data pengguna ganja yang dipublikasikan di *UCI Machine Learning* pada tahun 2016. Berdasarkan hasil pengujian dari penelitian yang telah dilakukan, didapatkan parameter optimal diantaranya 7 *hidden neuron* dan batas maksimal iterasi *K-Means* adalah 100. Dengan menggunakan parameter tersebut, didapatkan hasil akurasi sebesar 35,908%.

Kata kunci: ganja, klasifikasi, *Radial Basis Function Neural Network, K-Means, clustering*

ABSTRACT

Sukma Fardhia Anggraini, Determining Recency for Cannabis Consumption Using *Radial Basis Function Neural Network* (RBFNN)

Supervisors: Sigit Adinugroho, S.Kom, M. Sc and Randy Cahya Wihandika, S.ST, M.Kom.

In 2017, there are 1,742,285 cannabis (popular as marijuana) abusers in Indonesia. If a marijuana addict suddenly wants to stop using marijuana, it can cause symptoms of "sakau". To anticipate the symptoms of "sakau", rehabilitation treatment can be taken, so that marijuana addicts can get comprehensive treatment. Determining the appropriate type of rehabilitation, can make it useful. Then knowing the last time abusers had consumption the marijuana, be expected to provide supporting information to determine the appropriate rehabilitation program for marijuana addicts. One technique in data mining that can be used to solve this problem is classification techniques. In this study using Radial Radial Basis Function Neural Network (RBFNN) with K-Means as the classification method. The steps taken included data normalization, K-Means to found the value of centers and spread for Gaussian activation function, training and testing RBFNN. This study using 627 marijuana abuser data which was published on the UCI Machine Learning in 2016. The results of the research showed the optimal parameters involves 7 hidden neurons and 100 as the maximum limit of K-Means iterations. By using these parameters, the classification result achieved accuracy of 35,908%.

Keywords: *cannabis, marijuana, Radial Basis Function Neural Network, K-Means, classification, clustering*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar belakang

Merujuk pada Undang-Undang No. 35 Tahun 2009 tentang Narkotika pada Pasal 1 angka 1, Narkotika adalah zat atau obat yang dapat memberikan efek penurunan atau perubahan kesadaran, mengurangi atau menghilangkan rasa nyeri dan memiliki sifat yang adiktif serta berasal dari tanaman maupun bukan tanaman. Menurut Kepala Badan Narkotika Nasional (BNN), Komjen Budi Waseso, yang dikutip dari Kompas (2017), Indonesia menjadi negara darurat narkoba sejak tahun 1971. Berbagai upaya telah dilakukan pemerintah untuk mengurangi dan menanggulangi segala permasalahan narkoba, tetapi sesuai data yang dihimpun oleh BNN (2017), hingga tahun 2017 kasus peredaran maupun penyalahgunaan narkoba terus meningkat setiap tahunnya. Menurut hasil survei nasional yang dilakukan oleh Badan Narkotika Nasional (2017), bujukan teman dan rasa ingin mencoba menjadi alasan utama para pengguna mengonsumsi narkoba pada kali pertama. Padahal, sebanyak 91% responden mengetahui bahwa penyalahgunaan narkoba dapat mengakibatkan kerusakan otak.

Jenis narkoba yang paling banyak dikonsumsi oleh penyalahguna narkoba di Indonesia adalah ganja atau yang memiliki nama ilmiah *cannabis*, dengan jumlah penyalahguna mencapai 1.742.285 orang (Badan Narkotika Nasional, 2017). Berdasarkan Undang-Undang No. 22 Tahun 1997, ganja termasuk pada narkotika golongan I (satu). Menurut Humas BNN (2011), ganja merupakan tumbuhan berserat dengan kandungan zat narkotika tetrahidrokanabinol (THC) pada bijinya. Dengan adanya kandungan THC pada biji ganja, menyebabkan penggunanya dapat merasa gembira berlebihan tanpa sebab setelah mengonsumsinya. Konsumsi ganja dengan jangka panjang dapat memberikan dampak buruk terhadap kesehatan seperti radang paru-paru, iritasi, pembengkakan saluran nafas, kerusakan aliran darah koroner serta berisiko terhadap turunnya kadar hormon dan daya tahan tubuh. Selain dampak terhadap fisik, penyalahgunaan ganja juga berdampak terhadap psikis, seperti penurunan kemampuan untuk berbicara, berpikir, membaca maupun bergaul.

Dampak lain yang dapat terjadi ketika penyalahguna telah mengalami kecanduan ganja adalah sakau. Dikutip dari Hellosehat (2017), sakau dapat terjadi ketika pecandu obat menghentikan secara mendadak pemakaian obat tersebut maupun menurunkan secara drastis terhadap dosisnya sehingga dapat menimbulkan gejala tubuh sebagai respon terhadap perubahan pemakaian obat. Pada pecandu ganja, gejala sakau dialami sekitar 50% pecandu dengan pemakaian jangka panjang. Menurut Yayasan Pelita Ilmu (2017) untuk menanggulangi gejala sakau tidaklah mudah, sehingga rehabilitasi menjadi satu-satunya jalan terbaik agar pecandu dapat sembuh dari ketergantungan. Menurut Partodiharjo (2006), rehabilitasi narkoba adalah suatu metode pemulihan terhadap kesehatan jiwa maupun raga pecandu narkoba, sehingga dapat terbebas dari rasa candu terhadap narkoba dan terhindar dari dampak buruk yang ditimbulkan dari penyalahgunaan

narkoba. Rehabilitasi bisa dilakukan melalui rumah sakit ataupun panti rehabilitasi guna mendapatkan perawatan yang komprehensif. Mengutip pernyataan dr. Diah Setia Utami SpKJ, Ma, yang merupakan Deputi Rehabilitasi BNN, salah satu yang menentukan program rehabilitasi yang sebaiknya dilakukan terhadap pecandu adalah tingkat penggunaan narkoba. Menurut BNN Baddoka Makassar (2017), terdapat tiga program rehabilitasi, diantaranya rehabilitasi rawat jalan, rehabilitasi rawat inap 4 bulan dan rehabilitasi rawat inap 6 bulan. Program rehabilitasi rawat jalan dapat dilakukan dengan syarat tingkat penggunaan masih pada tahap ringan hingga sedang. Program rehabilitasi rawat inap 4 bulan dapat dilakukan dengan syarat tingkat penggunaan pada tahap teratur pakai. Dan bagi pengguna yang telah memasuki tahap ketergantungan, diharuskan menjalani program rehabilitasi rawat inap 6 bulan. Sehingga dengan mengetahui waktu terakhir penggunaan ganja, diharapkan dapat memberi informasi pendukung untuk menentukan program rehabilitasi yang sesuai untuk pecandu ganja.

Fehrman et al (2017) dalam penelitiannya menyebutkan bahwa terdapat beberapa faktor yang dapat dijadikan landasan untuk mengetahui waktu terakhir penggunaan ganja. Faktor-faktor tersebut diantaranya adalah umur, tingkat pendidikan yang terakhir ditempuh, *openness to experience* atau tingkat keterbukaan terhadap orang lain, *agreeableness* atau tingkat kemufakatan dengan orang lain, *conscientiousness* atau tingkat kesungguhan untuk mencapai tujuan dan *impulsivity*.

Metode yang dapat digunakan untuk mengetahui waktu terakhir pemakaian narkoba jenis ganja adalah dengan klasifikasi menggunakan Jaringan Saraf Tiruan. Salah satu jenis Jaringan Saraf Tiruan untuk klasifikasi adalah dengan *Radial Basis Function*. Pada penelitian yang dilakukan oleh Mirawanti et al (2012) dengan menggunakan objek Rumahtangga miskin Kota Pasuruan untuk melakukan perbandingan terhadap Metode Regresi Logistik Ordinal dengan Jaringan Saraf Tiruan Fungsi Radial, memberikan hasil akurasi klasifikasi sebesar 67,08% terhadap metode RBFNN dengan *K-Means* dan hanya 38% terhadap metode Regresi Logistic Ordinal. Tidak hanya dengan *K-Means*, terdapat pula penelitian menggunakan Fuzzy C-Means (FCM) untuk penentuan nilai center *Radial Basis Function* (RBF), yaitu penelitian yang dilakukan oleh Astuti et al. (2017). Pada penelitian tersebut, mendapatkan rata-rata akurasi sebesar 86,996% terhadap data uji dan nilai error sebesar 13,003%. Selain itu, penelitian yang dilakukan oleh Dillak (2012) menggunakan Jaringan RBF pada obyek *medical prescription* penyakit jantung, mampu memberikan hasil ketepatan diagnosa dan kesesuaian jenis obat sebesar 85%. Dan penelitian yang terkait dengan klasifikasi durasi pemakaian narkoba pernah dilakukan sebelumnya oleh Fehrman et al (2017). Pada penelitian tersebut metode klasifikasi yang digunakan adalah Decision Tree, Random Forest, KNN, Linear Discriminant Analysis, *Gaussian* Mixture, Probability Density Function Estimation, Regresi Logistic dan Naïve Bayes. Hasil yang didapatkan pada penelitian tersebut memiliki nilai rata-rata sensitivitas dan spesifisitas hingga 75% untuk narkoba jenis ganja.

Berdasarkan pada masalah-masalah yang dipaparkan di atas dan penelitian terdahulu yang relevan, maka penulis mengusulkan untuk melakukan penelitian dengan judul Klasifikasi Waktu Terakhir Penggunaan Ganja dengan Metode *Radial Basis Function Neural Network* (RBFNN). Dengan dilakukannya penelitian ini diharapkan dapat memberi manfaat untuk alat bantu pendukung keputusan jenis rehabilitas yang sesuai untuk pemakai ganja.

1.2 Rumusan masalah

Dalam penilitian ini terdapat dua masalah yang dirumuskan sebagai berikut:

1. Berapa parameter terbaik untuk metode *Radial Basis Function Neural Network* (RBFNN) untuk menentukan waktu terakhir penggunaan ganja?
2. Berapa tingkat akurasi penentuan waktu terakhir penggunaan ganja dengan metode *Radial Basis Function Neural Network* (RBFNN)?

1.3 Tujuan

Tujuan yang ingin dicapai dalam penelitian ini adalah sebagai berikut:

1. Mendapatkan parameter terbaik untuk metode *Radial Basis Function Neural Network* (RBFNN) dalam menentukan waktu terakhir penggunaan ganja.
2. Mendapatkan nilai akurasi dalam penerapan metode *Radial basis Function Neural Network* (RBFNN) dalam penentuan waktu terakhir penggunaan ganja.

1.4 Manfaat

Penelitian ini dilakukan dengan harapan memberikan manfaat untuk mengetahui waktu terakhir penggunaan ganja sehingga dapat membantu memutuskan jenis rehabilitasi yang sesuai untuk pemakai ganja.

1.5 Batasan masalah

Dalam rangka menghindari ruang lingkup pembahasan penelitian yang terlalu luas, maka ditentukan batasan permasalahan sebagai berikut:

1. Jenis narkoba yang digunakan adalah narkoba jenis ganja atau *cannabis sativa*.
2. Data latih dan data uji yang digunakan adalah data pemakaian narkoba dari UCI *Machine Learning*.
3. Bahasa pemrograman yang digunakan dalam penelitian adalah Java dengan IDE Netbeans 8.0.2.
4. Metode yang digunakan adalah *Radial basis Function Neural Network* (RBFNN) dengan *K-Means*.

1.6 Sistematika pembahasan

BAB 1 PENDAHULUAN

Pada bab 1, dibahas mengenai latar belakang dilakukannya peneilitian, rumusan masalah penelitian, batasan-batasan permasalahan yang dibahas, tujuan dilakukannya penelitian, manfaat dari dilakukannya penelitian dan sistematika pembahasan.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab 2, dibahas mengenai kajian pustaka dari penelitian-penelitian terdahulu dan dasar teori dari bermacam sumber yang terkait dengan judul penelitian.

BAB 3 METODOLOGI PENELITIAN

Pada bab 3, dibahas mengenai tahapan metode yang dilakukan pada penelitian. Terdiri dari tipe penelitian, strategi, metode pengumpulan data, metode analisis data dan rancangan penelitian.

BAB 4 PERANCANGAN

Pada bab 4, dibahas mengenai rancangan algoritma yang akan diimplementasikan ke dalam sistem. Pembahasan akan meliputi formulasi permasalahan dan strategi penyelesaian masalah.

BAB 5 IMPLEMENTASI

Pada bab 5, dibahas mengenai implementasi dari sistem yang telah dirancang pada bab sebelumnya. Terdapat *source code* dari sistem yang dibangun dan tampilan dari sistem.

BAB 6 PENGUJIAN DAN ANALISIS

Pada bab 6, dibahas mengenai pengujian mengenai metode *Radial Basis Function Neural Network* (RBFNN) untuk masalah klasifikasi waktu terakhir penggunaan ganja beserta dengan analsis dari pengujian.

BAB 7 PENUTUP

Pada bab 7, dibahas mengenai hal-hal yang dapat disimpulkan dari penelitian, meliputi akurasi sistem dan analisis serta saran untuk penelitian berikutnya.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Beberapa penelitian yang pernah dilakukan sebelumnya dan relevan dengan penelitian yang dilakukan saat ini, ditampilkan pada Tabel 2.1.

Tabel 2.1 Daftar Penelitian Terdahulu yang Relevan

No	Judul	Obyek	Metode	Hasil
1	*Perbandingan Metode Regresi Logistik Ordinal dengan Jaringan Saraf Tiruan Fungsi Radial Basis (Studi Kasus: Klasifikasi Rumahtangga Miskin Kota Pasuruan Tahun 2008) (Mirawanti et al, 2012)*	Rumahtangga miskin Kota Pasuruan hasil Program Pendataan dan Perlindungan Sosisal (PPLS) tahun 2008	Jaringan Saraf Tiruan Radial Basis Function dan Regresi Logistik Ordinal	Metode RBFNN dengan pendekatan *K-Means* mampu memberikan akurasi klasifikasi lebih tinggi dari metode Regresi Logistik Ordinal. Dengan pengujian *k-fold cross validation*, rata-rata nilai akurasi metode RBFNN dengan *K-Means* adalah 67,08% sedangkan Regresi Logistik Ordinal hanya 38%
2	*Penerapan Jaringan Saraf Tiruan Radial Basis Function pada Diaknosa dan Medical Prescription Penyakit Jantung (Dillak et al, 2012)*	Pasien dari Rumah Sakit Sahara dengan penyakit jantung yang dikumpulkan dari sesi OPD yang diperiksa oleh dokter setiap hari	Jaringan Saraf Tiruan Radial Basis Function	Dari 250 data yang digunakan sebagai data *training* dan 50 data sebagai data uji, sistem mampu memberikan hasil akurasi klasifikasi terhadap diagnosa penyakit jantung dan jenis obat yang sesuai sebesar 85%
3	*Lung Cancer Classification using Radial Basis Function Neural*	Foto X-ray jantung	Jaringan Saraf Tiruan Radial Basis Function	Dengan menerapkan Point Operation dalam jaringan RBF,

Tabel 2.1 Penelitian Terdahulu yang Relevan (lanjutan)

	Network Model with Point Operation (Wutsqa et al, 2017)		dengan Point Operation	sistem mampu mengklasifikan data dengan akurasi tertinggi untuk *training* sebesar 88.75% dan untuk *testing* 80%. Jumlah *hidden neuron* yang digunakan sebanyak 8 *neuron*.
4	*Analisis Leraning Jaringan RBF (Radial Basis Function Network) pada Pengenalan Pola Alfanumerik (Azmi, 2016)*	720 karakter alfanumerik dengan pola biner, terdiri dari numeric (0-9) dan alphabet capital (A-Z)	Jaringan Saraf Tiruan Radial Basis Function	Analisis dari hasil penelitian menyimpulkan bahwa Jaringan RBF mampu memberikan ketepatan pembelajaran sebesar 95%. Hal ini dikarenakan perhitungan matriks *Gaussian* yang menyebabkan proses perhitungan iterasi menjadi cepat
5	*Mammograms Classification using Gray-level Co-occurrence Matrix and Radial Basis Function Neural Network (Pratiwi et al, 2015)*	Data digital Mammogram yang dipublikasikan oleh MIAS database	Jaringan Saraf Tiruan Radial Basis Function dan Gray-level Co-occurrence Matrix	Dengan RBFNN hasil rata-rata akurasi klasifikasi yang didapatkan sebesar 91.57% dengan 84 hidden *neuron*.

2.2 Ganja

Ganja atau yang memiliki nama ilmiah *cannabis sativa*, merupakan salah satu jenis tumbuhan narkoba. Ganja digolongkan sebagai narkoba karena mampu memberikan efek euphoria atau rasa senang tanpa sebab kepada pemakainya karena terdapat kandungan zat Tetrahidrokanabinol (THC) di dalamnya. Penyalahgunaan ganja dapat menyebabkan berbagai efek berbahaya bagi tubuh pemakainya, baik secara fisik maupun secara psikis. Dan penyalahgunaan ganja

dengan dosis tinggi dapat memberikan dampak berupa peradangan pada paru-paru, iritasi hingga pembengkakan saluran nafas (Humas BNN, 2011).

2.2.1 Sakau Ganja

Ketika seorang pengguna telah menjadi pecandu, maka tidaklah mudah untuk menghilangkan rasa kecanduan terhadap ganja (Humas BNN, 2011). Sehingga ketika tiba-tiba menghentikan penggunaan ganja, maka akan muncul gejala sakau atau sakaw atau putus obat. Lebih lanjut tentang sakau, melansir dari situs kesehatan Hellosehat (2017), sakau terjadi ketika pecandu menghentikan pemakaian obat secara tiba-tiba atau menurukan dosis obat secara drastis sehingga menimbulkan gejala pada tubuh pecandu berat. Zat THC pada tanaman ganja akan berakibat langsung pada otak, sehingga otak para pecandu hanya dapat berfungsi normal ketika menggunakan ganja. Beberapa gejala fisik yang dirasakan oleh orang yang mengalami sakau ganja diantaranya mual, sakit perut, berkeringat, panas dingin dan gemetar.

2.2.2 Faktor yang Memengaruhi Tingkat Keparahan Sakau Ganja

Merujuk pada artikel di situs kesehatan Hellosehat (2017), beberapa faktor yang dapat memengaruhi tingkat keparahan gejala sakau ganja adalah sebagai berikut.

1. Tahap pemakaian ganja
2. Dosis pada tiap pemakaian ganja
3. Cara penggunaan ganja (dapat melalui hirupan dengan hidung, melalui kertas rokok ataupun ditelan langsung)
4. Riwayat keluarga dan genetik
5. Faktor kesehatan pecandu

2.2.3 Faktor yang Berpengaruh Terhadap Waktu Terakhir Penggunaan Ganja

Menurut penelitian yang dilakukan oleh Fehrman et al (2017), terdapat enam faktor yang dapat dijadikan landasan untuk mengetahui waktu terakhir penggunaan ganja, diantaranya:

1. Umur

   Merujuk pada Kamus Besar Bahasa Indonesia (KBBI) umur atau usia adalah seberapa lama seseorang hidup sejak dilahirkan, yang umumnya dinyatakan dalam satuan tahun.

2. Tingkat pendidikan yang terakhir ditempuh
3. *Openness to experience* (Oscore)

   Oscore merupakan tingkat keterbukaan seseorang terhadap orang lain. Orang dengan nilai oscore yang tinggi cenderung memiliki pemikiran yang imajinatif (Friedman dan Schustack, 2012).

4. *Agreeableness* (Ascore)

   Ascore merupakan tingkat kemufakatan seseorang terhadap orang lain. Seseorang dengan nilai ascore yang tinggi akan cenderung mudah mempercayai orang lain, murah hati, mudah menerima pendapat orang lain, mudah mengalah dan menghindari konflik (Feist dan Feist, 2009).

5. *Conscientiousness* (Cscore)

   Cscore merupakan tingkat kesungguhan seseorang terhadap tujuan yang ingin dicapai. Seseorang dengan nilai cscore yang tinggi cenderung dapat diandalkan, teratur, fokus terhadap tujuan dan bertanggung jawab. Sebaliknya, jika nilai cscore seseorang rendah, maka ia cenderung ceroboh, berantakan, tidak terarah, mudah teralih perhatiannya dan tidak dapat diandalkan (Friedman dan Schustack, 2012).

6. *Impulsivity*

   Merupakan tingkat dorongan dalam diri untuk melakukan tindakan untuk kepuasan yang ingin dicapai, baik secara sadar maupun tidak sadar. Salah satu metode yang dapat digunakan untuk mengukur tingkat *impulsivity* sesorang adalah dengan BIS-11 (Patton et al, 1995).

2.3 Rehabilitasi

Bagi pecandu ganja, salah satu cara yang dapat digunakan untuk mengurangi gejala sakau dan menghilangkan rasa kecanduan terhadap narkoba, adalah melalui rehabilitasi. Merujuk pada Kamus Besar Bahasa Indonesia (KBBI), rehabilitasi adalah pemulihan kepada kedudukan yang dahulu (semula). Sedangkan pengertian rehabilitasi narkoba menurut Partodiharjo (2006) adalah suatu metode pemulihan terhadap kesehatan jiwa maupun raga pecandu narkoba, sehingga dapat terbebas dari rasa candu terhadap narkoba dan terhindar dari dampak buruk yang ditimbulkan dari penyalahgunaan narkoba. Mengutip pernyataan dr. Diah Setia Utami SpKJ, Ma selaku Deputi Rehabilitasi BNN, jenis program rehabilitasi dapat ditentukan salah satunya berdasarkan pada tingkat penggunaan narkoba (AntaraNews, 9 Januari 2015). Menurut BNN Baddoka Makassar (2017), terdapat tiga program rehabilitasi, diantaranya sebagai berikut:

1. Rehabilitasi Rawat Jalan

   Jika pengguna narkoba masih berada pada tingkat penggunaan ringan hingga sedang.

2. Rehabilitasi Rawat Inap 4 Bulan

   Jika pengguna narkoba berada pada tingkat penggunaan teratur pakai. Selain itu, telah terjadi komplikasi kesehatan fisik dan psikis, tetapi masih pada tingkat ringan hingga sedang.

3. Rehabilitasi Rawat Inap 6 Bulan

   Jika pengguna narkoba telah menggunakan narkoba secara rutin pada jangka waktu yang relatif lama, sehingga telah mencapai tingkat ketergantungan. Serta telah terjadi komplikasi berat pada kesehatan fisik dan psikis.

2.4 Normalisasi *Min-Max*

Normalisasi *min-max* merupakan salah satu metode normalisasi data dengan konsep merubah skala data dalam rentang 0 dan 1 (Chamidah et al., 2012). Untuk merubah data ke dalam rentang baru 0 dan 1, maka dilakukan normalisasi dengan Persamaan 2.1.

$$x'_{ij} = \frac{x_{ij} - min_j}{max_j - min_j} \quad (2.1)$$

Keterangan:

x'_{ij} = Nilai hasil normalisasi data ke-i fitur ke- j

x_{ij} = Nilai data ke-i fitur ke- j

min_j = Nilai minimum dari fitur ke- j

max_j = Nilai maksimum dari fitur ke- j

2.5 Metode *K-means*

Menurut Kuntjoro et al (2018), *clustering* adalah penglompokan data yang bertujuan untuk mencari data dengan karakteristik yang sama untuk dijadikan satu klaster dan memisahkan klaster untuk data dengan karakteristik yang berbeda. *Clustering* juga disebut sebagai *unsupervised classification* karena mengelompokkan tidak berdasarkan kelas yang sudah ada, tetapi berdasarkan kesamaan karakteristik data. Salah satu metode yang menerapkan konsep *clustering* adalah *K-Means*. Menurut Hruschka dan Ebecken (dalam Kuntjoro et al, 2018), *K-Means* menerapkan metode *non-hierarchial* dalam penelompokan data untuk membagi data dalam klaster-klaster.

Langkah-langkah metode *K-Means* adalah sebagai berikut.

1. Inisialisasi variabel yang diperlukan, yaitu: jumlah klaster, nilai fungsi obyektif awal, *threshold*, iterasi maksimum dan klaster awal setiap data.
2. Hitung *centroid* atau pusat klaster menggunakan Persamaan 2.2

   $$C_{kj} = \frac{1}{N_k} \sum_{k=0}^{N_k} X_{ij} \quad (2.2)$$

   Keterangan:

   C_{kj} = *centroid* atau nilai *mean* klaster ke-k untuk fitur ke-j

   N_k = jumlah data pada klaster ke-k

   X_{ij} = nilai data ke-i untuk fitur ke-j

3. Hitung jarak tiap data dengan pusat klaster *(centroid)*. Perhitungan jarak dilakukan menggunakan metode *Euclidean Distance* dengan rumus pada Persamaan 2.3.

$$d_{ik} = \sqrt{\sum_{j=1}^{p}(X_{ij} - C_{kj})^2} \quad (2.3)$$

Keterangan:

d_{ik} = jarak antara data ke-i dengan klaster ke-k

P = jumlah fitur

X_{ij} = nilai data ke-i pada fitur ke-j

C_{kj} = nilai centroid klaster ke-k pada fitur ke-j

4. Perbarui klaster setiap data

Dari perhitungan sebelumnya, cari jarak minimum untuk tiap data pada semua klaster. Jarak minimum itulah yang menjadi dasar penentuan klaster terbaru untuk data tersebut.

5. Hitung fungsi obyektif terbaru.

Nilai fungsi obyektif didapatkan dengan menjumlahkan seluruh nilai jarak minimal data ke klaster dengan Persamaan 2.4.

$$F_t = \sum_{i=0}^{n} d_{min}^{ik} \quad (2.4)$$

Keterangan:

F_t = nilai fungsi objektif iterasi ke- t

n = jumlah data

d_{min}^{ik} = jarak minimum data ke- i ke semua klaster k

6. Hitung nilai delta fungsi objektif dengan Persamaan 2.5

$$\Delta F_t = |F_t - F_{t-1}| \quad (2.5)$$

Keterangan:

ΔF_t = delta fungsi objektif pada iterasi ke-t

F_t = nilai fungsi objektif pada iterasi ke-t

F_{t-1} = nilai fungsi objektif pada iterasi ke-$(t-1)$

7. Ulangi iterasi mulai dari langkah 4-6 jika kriteria berhenti tidak terpenuhi. Berikut kriteria berhenti pada *K-Means*:

   a. Jika nilai iterasi sama dengan nilai maksimum iterasi

   b. Jika nilai *threshold* lebih besar dari delta fungsi obyektif

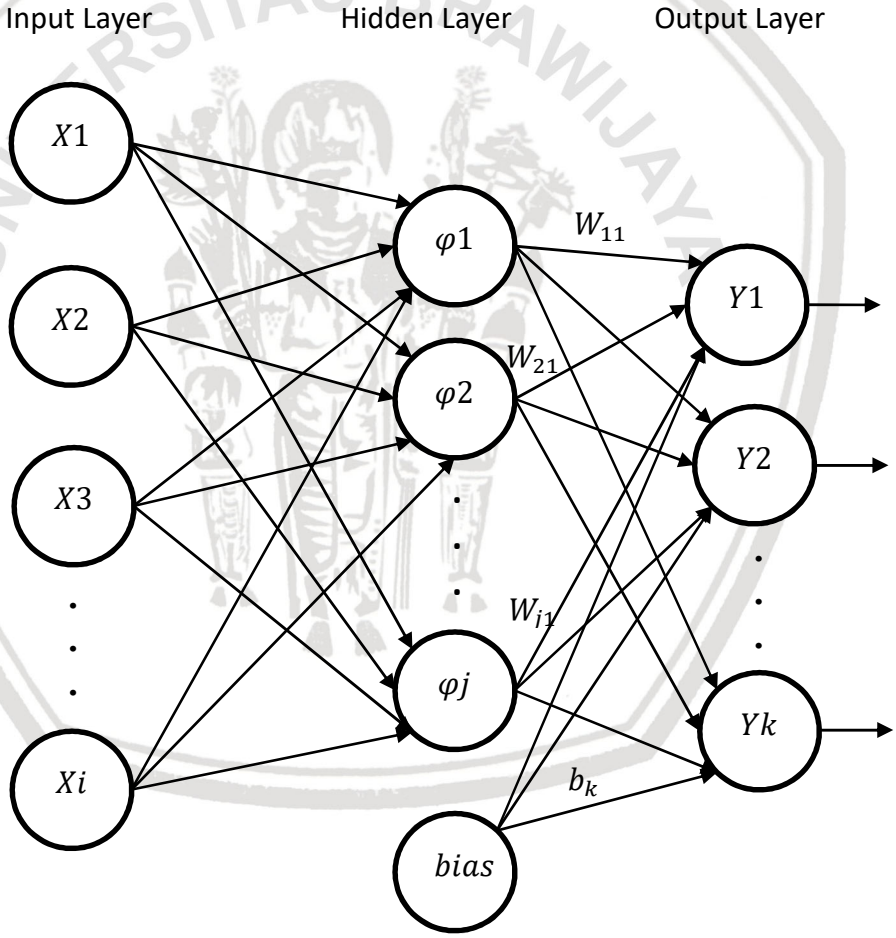
2.6 Jaringan Saraf Tiruan *Radial Basis Function*

Menurut Azmi (2016), jaringan saraf tiruan *Radial Basis Function* (RBF) merupakan jaringan yang memiliki kemiripan dengan model jaringan saraf tiruan *Multilayer Perceptron* (MLP *network*). Pada jaringan RBF, *hidden layer* bersifat non-linier dengan menggunakan fungsi aktivasi *Gaussian* dan *output layer* bersifat linier sehingga menggunakan fungsi aktivasi linier.

Pada jaringan RBF, dari *input layer* ke *hidden layer* tidak menerapkan operasi perkalian matriks antara data *input* dengan bobot, tetapi dengan menghitung jarak antara data input dengan nilai *center* dari setiap *hidden neuron* (Azmi, 2016). Nilai *center* dari *hidden neuron* dapat ditentukan salah satunya dengan metode *unsupervised* seperti algoritme *K-means* (Santosa, 2016).

2.6.1 Arsitektur Jaringan Saraf Tiruan *Radial Basis Function*

Arsitektur jaringan RBF ditampilkan pada Gambar 2.1



Gambar 2.1 Arsitektur Jaringan Saraf Tiruan *Radial Basis Function*.

2.6.2 *Radial Basis Function Neural Network* dengan *K-Means*

Pada metode RBFNN dengan *K-Means*, dilakukan dua langkah utama. Langkah pertama adalah *clustering* data dengan *K-Means*. Pada langkah pertama bertujuan

untuk mendapatkan nilai *mean* tiap atribut pada tiap klaster atau *hidden neuron*, nilai *spread*, bobot dan bias. Setelah mendapatkan hasil *training* maka langkah kedua adalah melakukan klasifikasi dengan RBFNN (Mirawanti et al., 2012).

Langkah-langkah metode *training* pada RBFNN adalah sebagai berikut.

1. Inisialisasi nilai *centroid* tiap klaster menggunakan metode *K-Means*.
2. Menghitung nilai *spread* σ. Nilai *spread* untuk fungsi *Gaussian* dapat menggunakan nilai yang sama untuk semua *hidden neuron*, dengan Persamaan 2.6 (Haykin, 2009).

   $$\sigma_j = \frac{d_{max}}{\sqrt{2K}} \quad (2.6)$$

   Keterangan:

   σ_j = nilai *spread hidden neuron* ke- j

   d_{max} = jarak maksimum dari pusat klaster yang terbentuk

   K = banyaknya klaster yang terbentuk
3. Meneruskan sinyal dari *input layer* ke *hidden layer* dengan fungsi aktivasi *Gaussian* menggunakan rumus pada Persamaan 2.7.

   $$\varphi_j = exp\left(-\frac{\|X_i - C_j\|^2}{2\sigma_j^2}\right) \quad (2.7)$$

   Keterangan:

   φ_j = fungsi *Gaussian* pada *hidden neuron* ke-j

   $\|X_i - C_j\|^2$ = jarak antara data ke- i dengan nilai *centroid* dari *hidden neuron* ke- j

   σ_j^2 = nilai *spread hidden neuron* ke- j
4. Menyusun matrik *Gaussian* dari langkah 3 seperti ditampilkan pada Persamaan 2.8 dan kolom terakhir diisi dengan nilai bias.

   $$G = [\varphi_1 \dots \varphi_j \; b] \quad (2.8)$$

   Keterangan:

   G = matriks *Gaussian*

   j = indeks *hidden neuron*

   b = bias
5. Melakukan *update* bobot dengan *pseudoinverse* dari matriks *Gaussian* dan perkalian dengan target menggunakan Persamaan 2.9.

   $$W = (G^T G)^{-1} G^T t \quad (2.9)$$

   Keterangan:

   W = matriks bobot dari *hidden neuron* ke *output neuron*

G^T = *transpose* dari matrik G

$(G^T G)^{-1}$ = *inverse* dari perkalian G^T dan G

t = matriks target

6. Menyimpan hasil dari proses *training* yang berupa matrik *centroid*, *spread*, bobot dan bias untuk proses *testing*.

Langkah-langkah metode *testing* pada RBFNN adalah sebagai berikut:

1. Menghitung matriks *Gaussian* untuk data uji menggunakan Persamaan 2.7 hingga Persamaan 2.8 dengan nilai *centroid* dan nilai *spread* yang didapat dari hasil *training*.
2. Menghitung output jaringan RBF dengan Persamaan 2.10. Nilai w_{jk} dari matriks bobot W dan nilai bias didapatkan dari proses *training*.

$$y_k = \sum_{j=1}^{m} \varphi_{ij} w_{jk} + b_k \quad (2.10)$$

Keterangan:

y_k = nilai keluaran pada *output neuron* ke-k

w_{jk} = bobot dari *hidden neuron* ke-j ke *output neuron* ke-k

φ_{ij} = fungsi aktivasi *Gaussian* data ke-i *hidden neuron* ke-j

b_k = bobot bias untuk *output neuron* ke-k

3. Menentukan kelas setiap data dari indeks *output neuron* yang memiliki nilai y_k maksimum untuk data tersebut.

2.7 Evaluasi

Untuk mengetahui nilai akurasi dari hasil klasifikasi sistem, maka dilakukan perhitungan akurasi dengan Persamaan 2.11.

$$akurasi = \frac{jumlah\ data\ benar}{jumla\quad data}\ x\ 100\% \quad (2.11)$$

Pada penelitian ini, metode evaluasi data yang digunakan adalah *K-Fold Cross Validation*. Pada *k-fold cross validation*, terdapat dua jenis data untuk dilakukan evaluasi, yaitu data latih dan data uji. Data akan dibagi menjadi beberapa lipatan atau *fold*, kemudian setiap *fold* secara bergantian akan menjadi data latih dan data uji. Ketika satu *fold* menjadi data uji, maka sisa *fold* yang lain akan menjadi data latih, begitu seterusnya hingga semua *fold* telah menjadi data uji maupun data latih (Rafaeilzadeh et al., 2009).

BAB 3 METODOLOGI

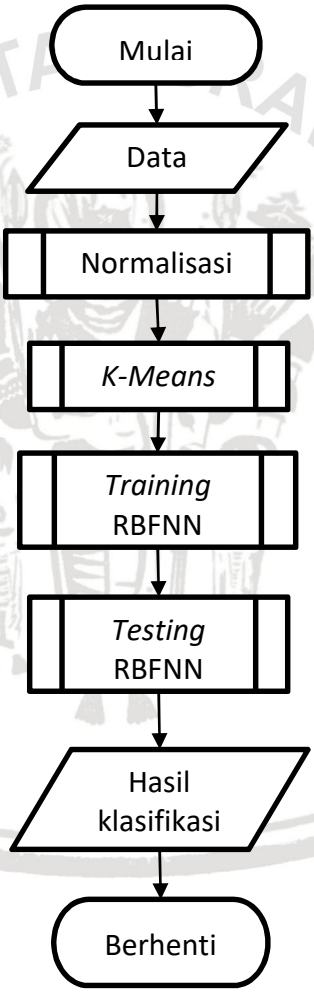
3.1 Tipe penelitian

Penelitian ini bertujuan menganalisis metode RBFNN dengan *K-Means* untuk menentukan waktu terakhir penggunaan ganja. Oleh karena itu, penelitian ini merupakan tipe penelitian nonimplementatif analitik.

3.2 Strategi dan Rancangan Penelitian

3.2.1 Metode Secara Umum

Metode yang digunakan dalam penelitian ini, direpresentasikan dalam Gambar 3.1.



Gambar 3.1 Metode Secara Umum

Metode yang dilakukan pada penelitian ini diawali dengan memasukkan data yang akan diklasifikasi. Data terbagi menjadi dua, yaitu data latih dan data uji. Kemudian data akan dinormalisasi dan dikelompokkan menggunakan *K-Means* untuk menemukan nilai *center* dan nilai *spread* pada *hidden layer*. Langkah

berikutnya adalah melakukan pelatihan RBFNN untuk mendapatkan bobot dan bobot bias dari *hidden layer* menuju ke *output layer*. Setelah mendapatkan bobot, maka dilakukan proses terakhir yaitu proses *testing* dengan data uji hingga mendapatkan hasil klasifikasi.

3.2.2 Lokasi Penelitian

Penelitian dilakukan di Laboratorium Riset Fakultas Ilmu Komputer, Universitas Brawijaya.

3.2.3 Metode Pengumpulan Data

Pada penelitian ini, data yang digunakan adalah data sekunder dari *UCI Machine Learning* dengan judul *Drug Consumption* yang diterbitkan pada tahun 2016 sebanyak 627 data (Fehrman et al., 2016). Dalam data tersebut terdapat 6 fitur, diantaranya: Umur, pendidikan terakhir, *Oscore*, *Ascore*, *Cscore* dan *Impulsivity* (BIS-11). Kemudian terdapat 7 kelas waktu terakhir penggunaan ganja, diantaranya: CL0 (tidak pernah menggunakan), CL1 (menggunakan di atas 10 tahun lalu), CL2 (menggunakan pada 10 tahun terakhir), CL3 (menggunakan pada 1 tahun terakhir), CL4 (menggunakan pada 1 bulan terakhir), CL5 (menggunakan pada 1 minggu terakhir) dan CL6 (menggunakan pada 1 hari sebelumnya).

3.2.4 Metode Analisis Data

Pada penelitian ini, data yang telah diklasifikasi menggunakan RBFNN akan dianalisis menggunakan pengujian akurasi. Pengujian akurasi akan mengukur seberapa sesuai kelas hasil keluaran sistem dengan kelas data yang sesungguhnya. Pengujian akurasi dilakukan dengan menerapkan Persamaan 2.11.

3.2.5 Peralatan Pendukung

Pada penelitian ini, daftar peralatan pendukung yang digunakan adalah sebagai berikut.

1. *Hardware*
    - Laptop HP 14R017TX
    - Prosesor Intel Core i3-4030U
    - RAM 4 GB
2. *Software* dan *Tools*
    - Sistem Operasi Windows 10 64-bit
    - Bahasa pemrograman Java
    - IDE NetBeans 8.0
    - Library JAMA 1.0.3
    - Microsoft Excel 2013

BAB 4 PERANCANGAN

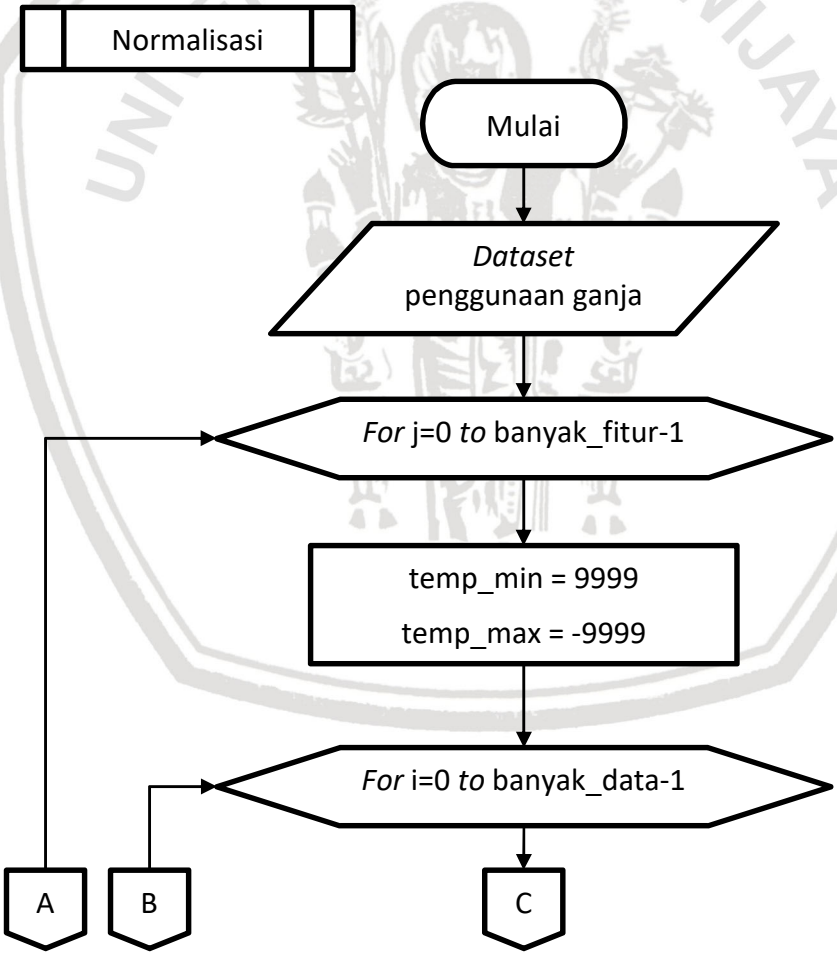
4.1 Formulasi Permasalahan

Pada penelitian ini, permasalahan yang akan diselesaikan adalah menentukan waktu terakhir penggunaan ganja dengan metode *Radial Basis Function Neural Network* (RBFNN). Data yang digunakan adalah 627 data pengguna ganja yang didapatkan dari *UCI Machine Learning* dengan topik *Drug Consumption* yang diterbitkan pada tahun 2016. Hasil klasifikasi dievaluasi menggunakan metode pengujian *k-fold cross validation*.

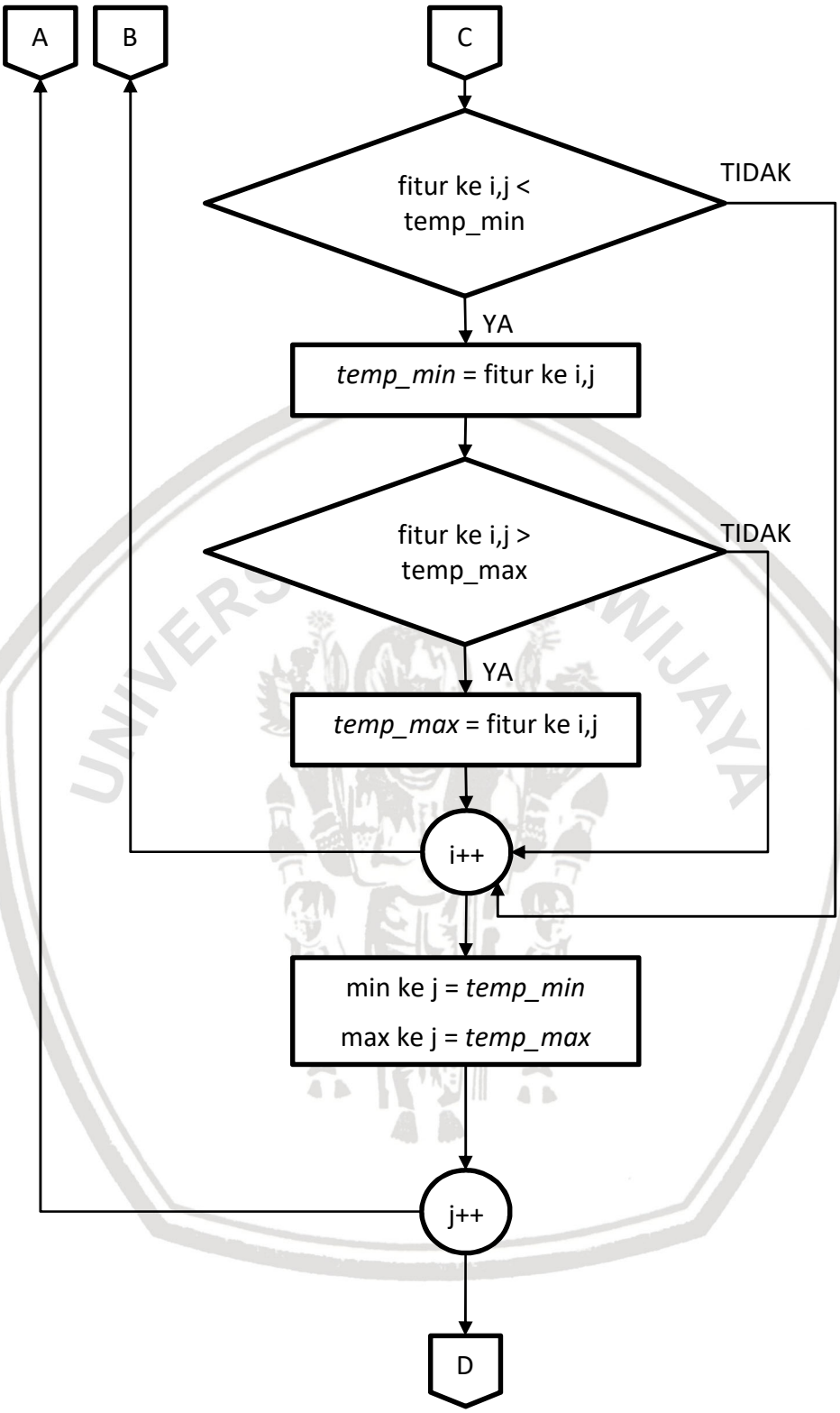
4.2 Strategi Penyelesaian Permasalahan

4.2.1 Normalisasi

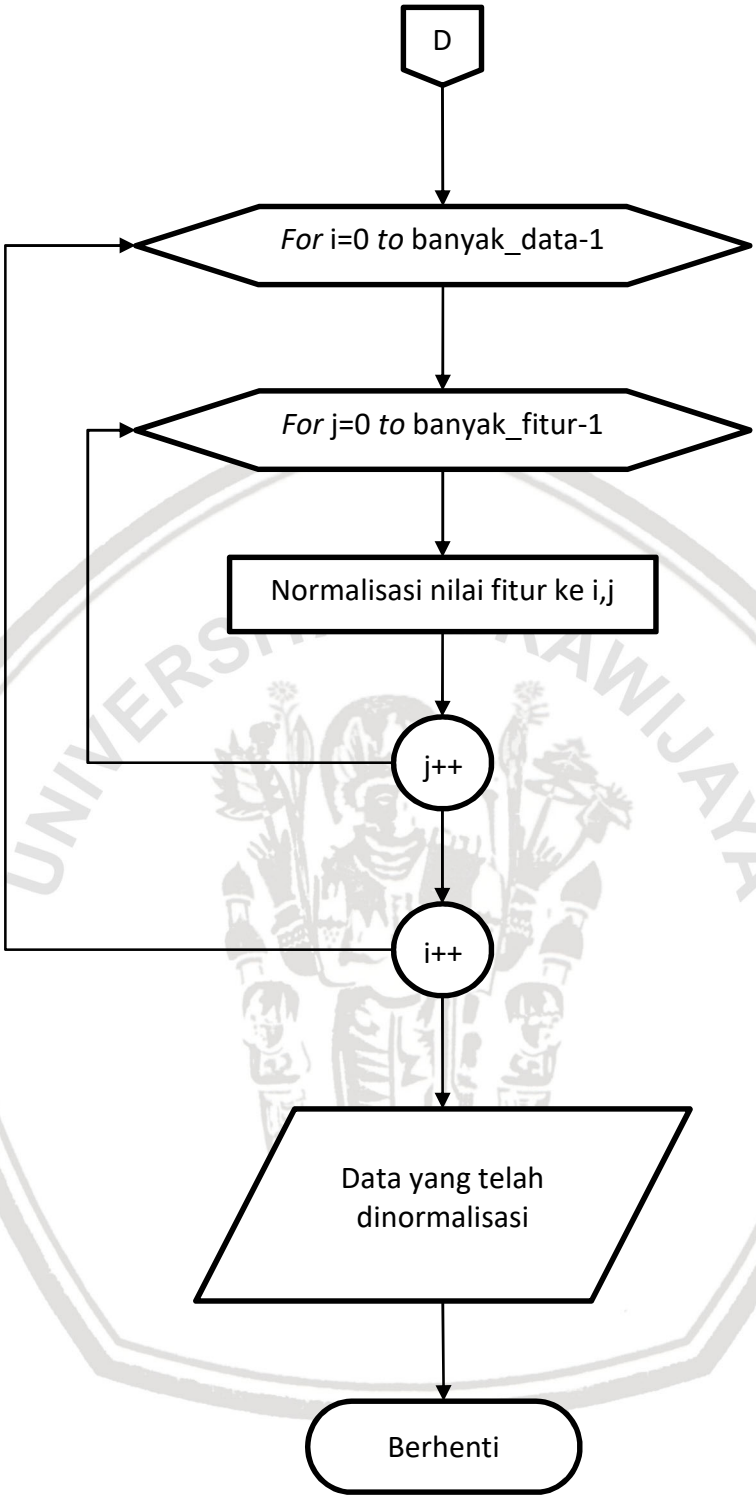
Proses normalisasi data berfungsi untuk menormalkan nilai ke dalam rentang 0 dan 1. Diagram alir *K-Means* ditampilkan pada Gambar 4.1, Gambar 4.2 dan Gambar 4.3.



Gambar 4.1 Diagram Alir Normalisasi (Bagian 1)



Gambar 4.2 Diagram Alir Normalisasi (Bagian 2)



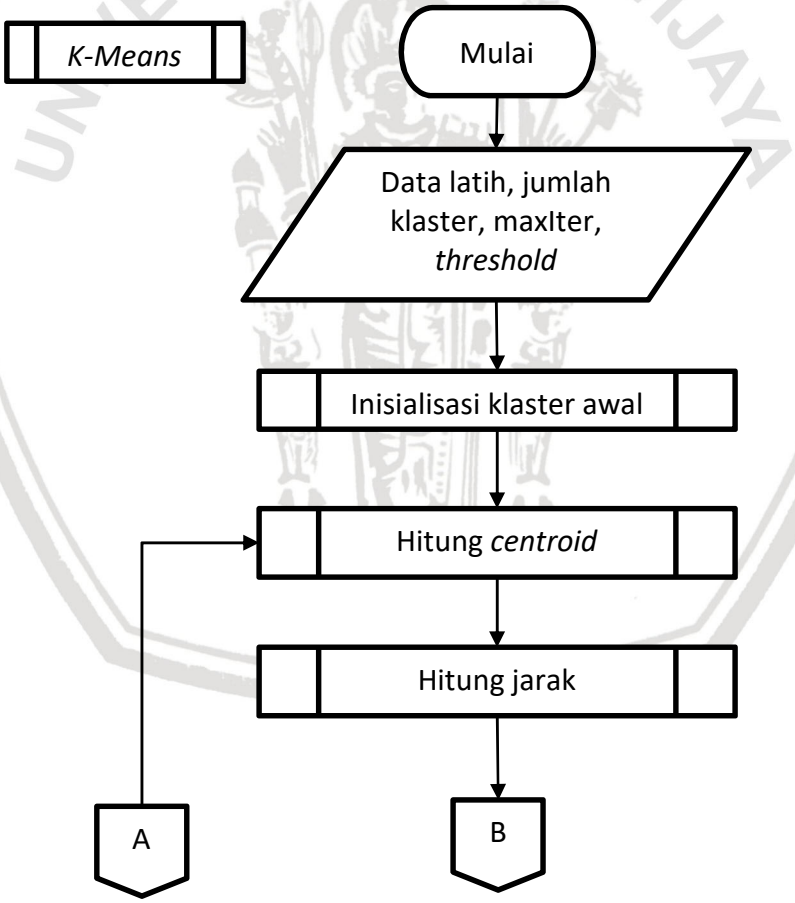
Gambar 4.3 Diagram Alir Normalisasi (Bagian 3)

Penjelasan dari diagram alir proses inisialisasi normalisasi sebagai berikut.

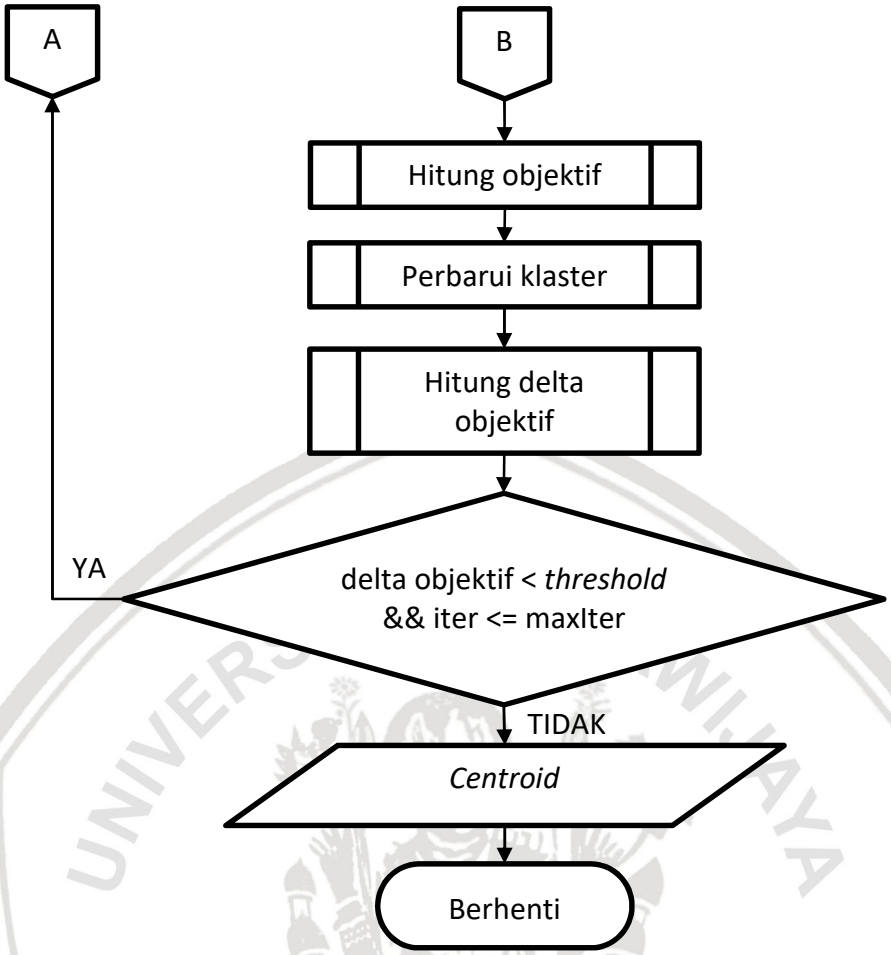
1. Membentuk matriks yang berisi data masukan, yaitu data konsumsi narkoba. Matriks yang terbentuk memiliki dimensi baris sejumlah data dan dimensi kolom sejumlah fitur ditambah kolom terakhir berisi kelas dari data.
2. Mencari nilai minimum dan maksimum dari setiap fitur, kemudian disimpan ke dalam *array* 'min' untuk nilai minimum dan *array* 'max' untuk nilai maksimum.
3. Melakukan normalisasi *min-max* dengan Persamaan 2.1 untuk semua data.
4. Menyimpan data hasil normalisasi ke dalam *array* 'fitur'.

4.2.2 *K-Means*

Proses *K-Means* terdiri dari beberapa sub proses di dalamnya. *K-Means* menerima masukan berupa data yang telah dinormalisasi, jumlah klaster, maksimum iterasi dan *threshold*. Dan keluaran yang dihasilkan adalah berupa *centroid* dan nomor klaster setiap data pada iterasi terakhir. Diagram alir *K-Means* ditampilkan pada Gambar 4.4 dan Gambar 4.5.



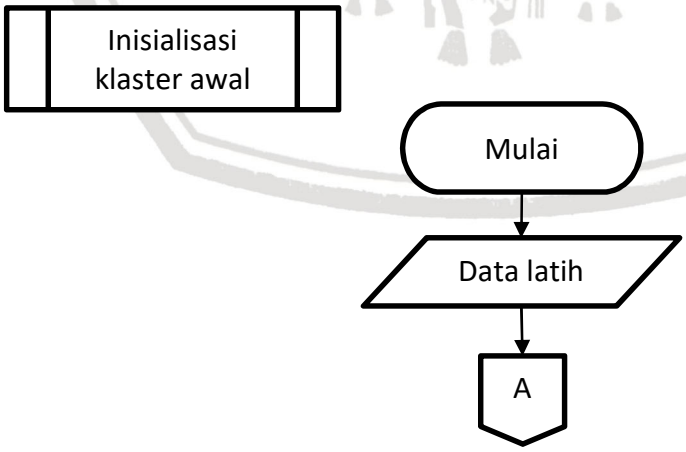
Gambar 4.4 Diagram Alir *K-Means* (Bagian 1)



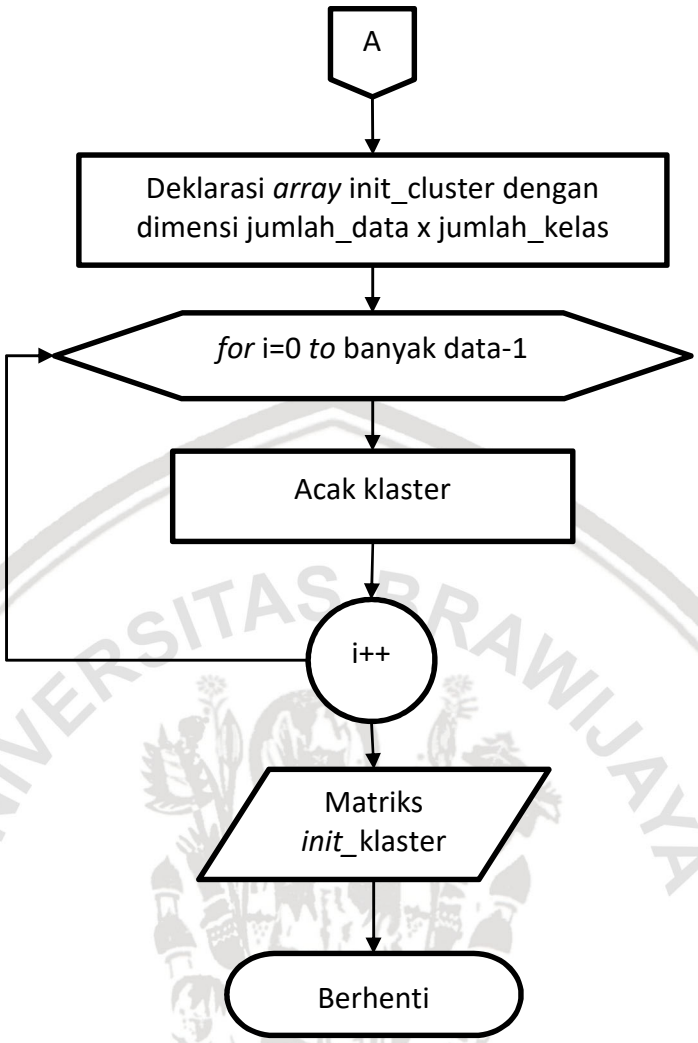
Gambar 4.5 Diagram Alir *K-Means* (Bagian 2)

4.2.2.1 Inisialisasi Klaster Awal

Pada penelitian ini, inisialisasi nomor klaster dilakukan secara acak. Diagram alir proses inisialisasi klaster ditampilkan pada Gambar 4.6 dan Gambar 4.7.



Gambar 4.6 Diagram Alir Inisialisasi Klaster Awal (Bagian 1)



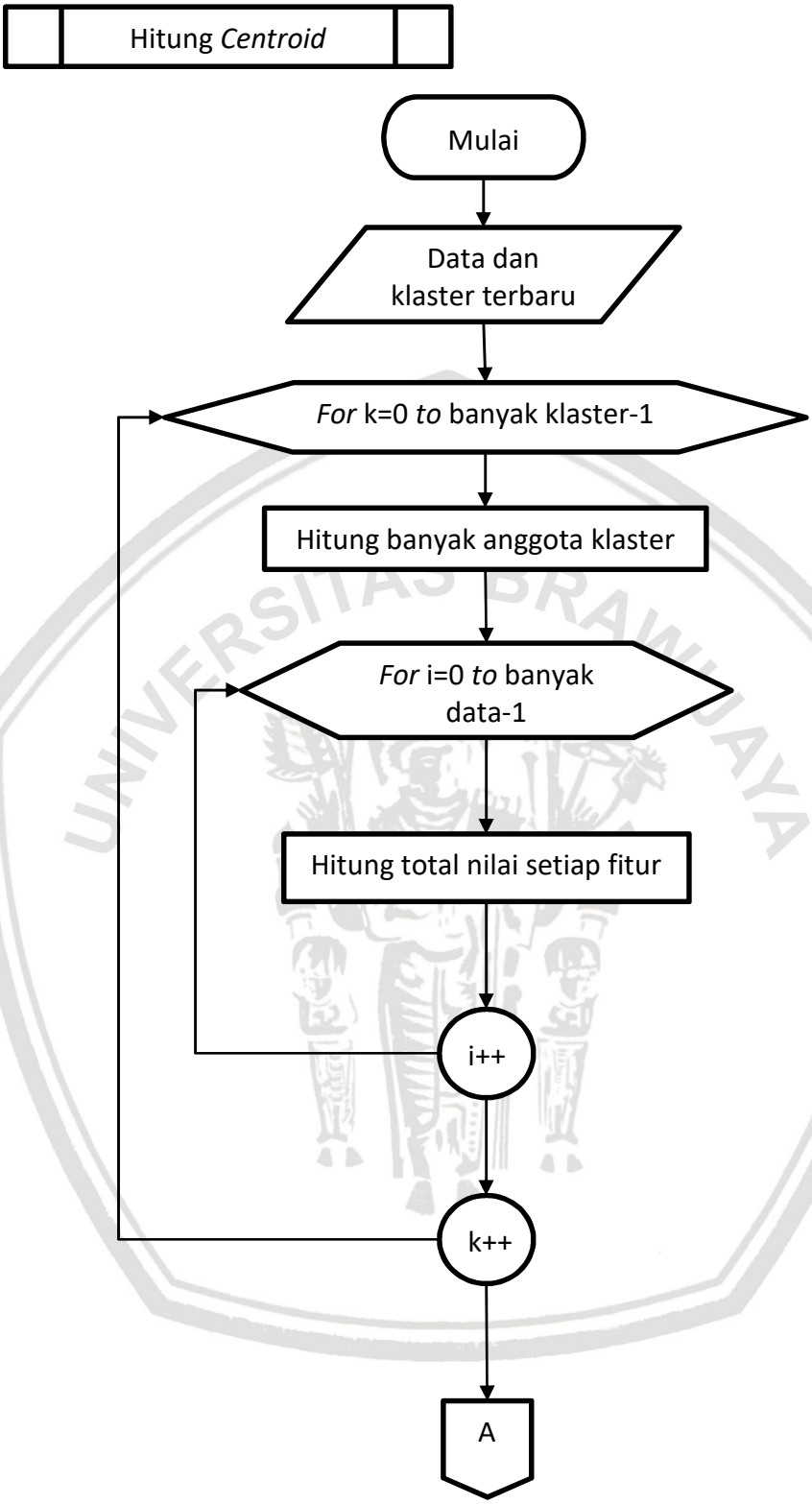
Gambar 4.7 Diagram Alir Inisialisasi Klaster Awal (Bagian 2)

Penjelasan dari diagram alir proses inisialisasi klaster adalah sebagai berikut.

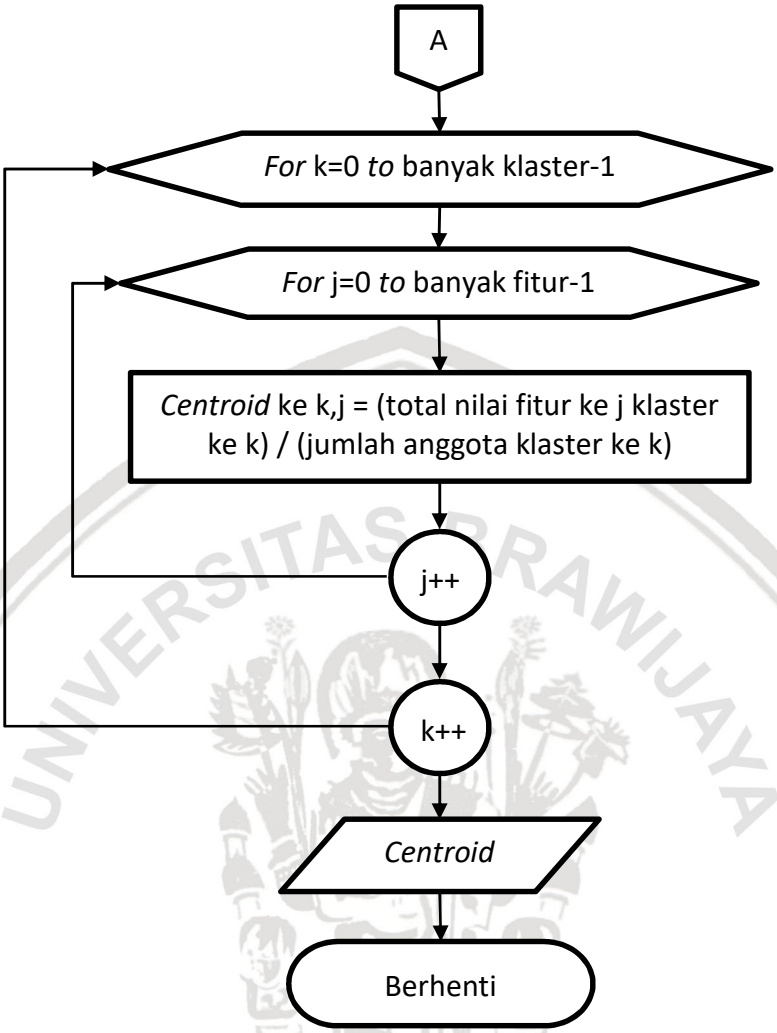
1. Membentuk matriks yang berisi data masukan, yaitu data konsumsi narkoba. Matriks yang terbentuk memiliki dimensi baris sejumlah data dan dimensi kolom sejumlah fitur ditambah kolom terakhir berisi kelas dari data.
2. Mendeklarasikan matriks init_cluster yang akan menyimpan letak klaster setiap data.
3. Menyimpan matriks init_cluster untuk proses selanjutnya.

4.2.2.2 Hitung *Centroid*

Proses untuk menghitung *centroid* klaster menggunakan data dan klaster terbaru setiap data. *Centroid* akan selalu diperbarui pada setiap iterasi hingga kriteria berhenti terpenuhi. *Centroid* digunakan untuk menentukan letak klaster yang terdekat dengan kriteria data. Diagram alir proses untuk menghitung *centroid* ditampilkan pada Gambar 4.8 dan Gambar 4.9.



Gambar 4.8 Diagram Alir Hitung *Centroid* (Bagian 1)



Gambar 4.9 Diagram Alir Hitung *Centroid* (Bagian 2)

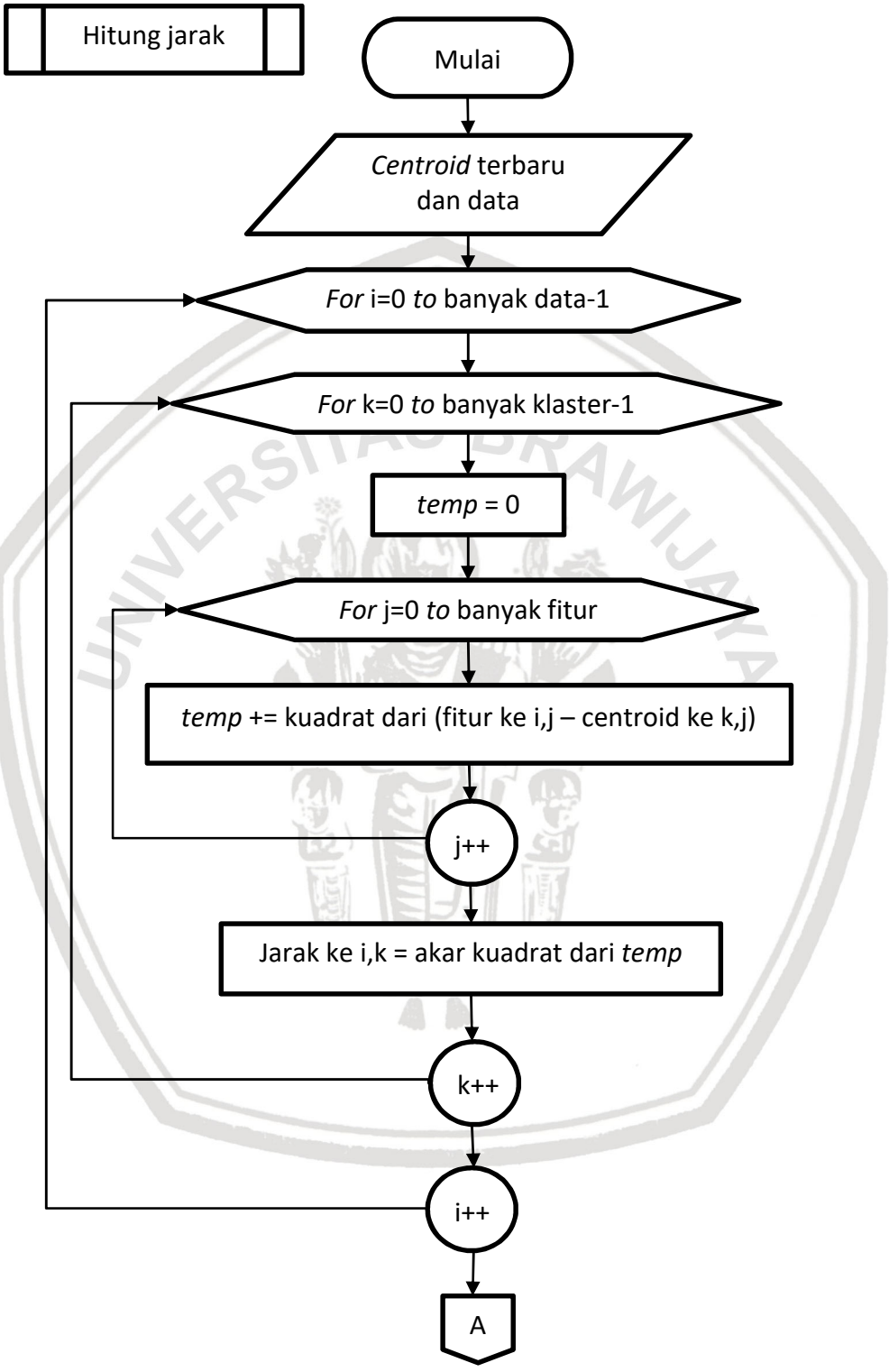
Penjelasan dari diagram alir proses menghitung *centroid* klaster adalah sebagai berikut.

1. Membentuk matriks data dan matriks letak klaster setiap data.
2. Menjumlahkan nilai setiap fitur dari data yang berada pada klaster yang sama dan menghitung jumlah data yang berada pada klaster yang sama.
3. Menghitung nilai rata-rata dari setiap fitur pada setiap klaster dengan menggunakan Persamaan 2.2.
4. Menyimpan hasil rata-rata setiap fitur dari setiap klaster ke dalam matriks *centroid* yang berdimensi baris sejumlah klaster dan dimensi kolom sejumlah fitur data.

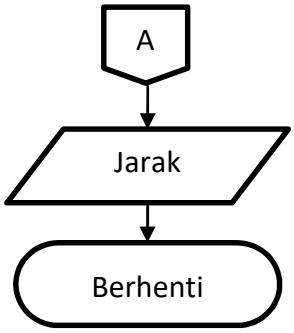
4.2.2.3 Proses Hitung Jarak

Untuk mengetahui pusat klaster yang paling dekat dengan data, maka perlu dilakuan perhitungan jarak antara data dengan *centroid* atau pusat klaster. Proses

perhitungan jarak yang dilakukan dalam penelitian ini menggunakan metode *Euclidean Distance*. Diagram alir proses untuk menghitung jarak data dan *centroid* setiap klaster ditampilkan pada Gambar 4.10 dan 4.11.



Gambar 4.10 Diagram Alir Proses Hitung Jarak (Bagian 1)



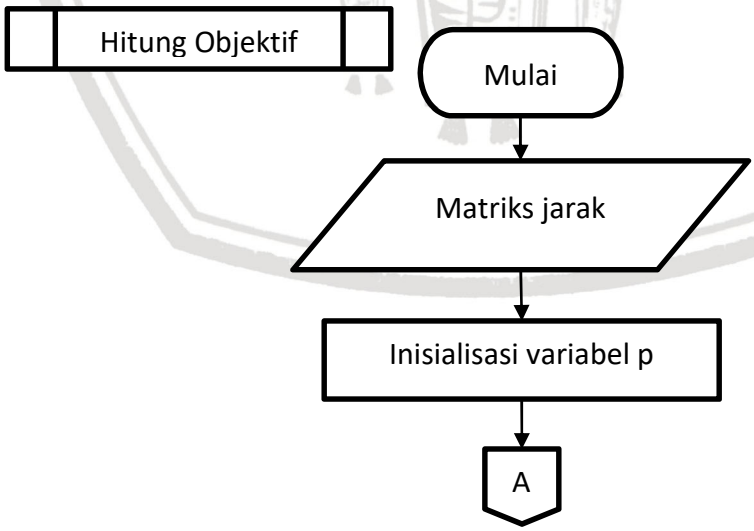
Gambar 4.11 Diagram Alir Hitung Jarak (Bagian 2)

Penjelasan dari diagram alir proses menghitung jarak antara klaster dengan data adalah sebagai berikut.

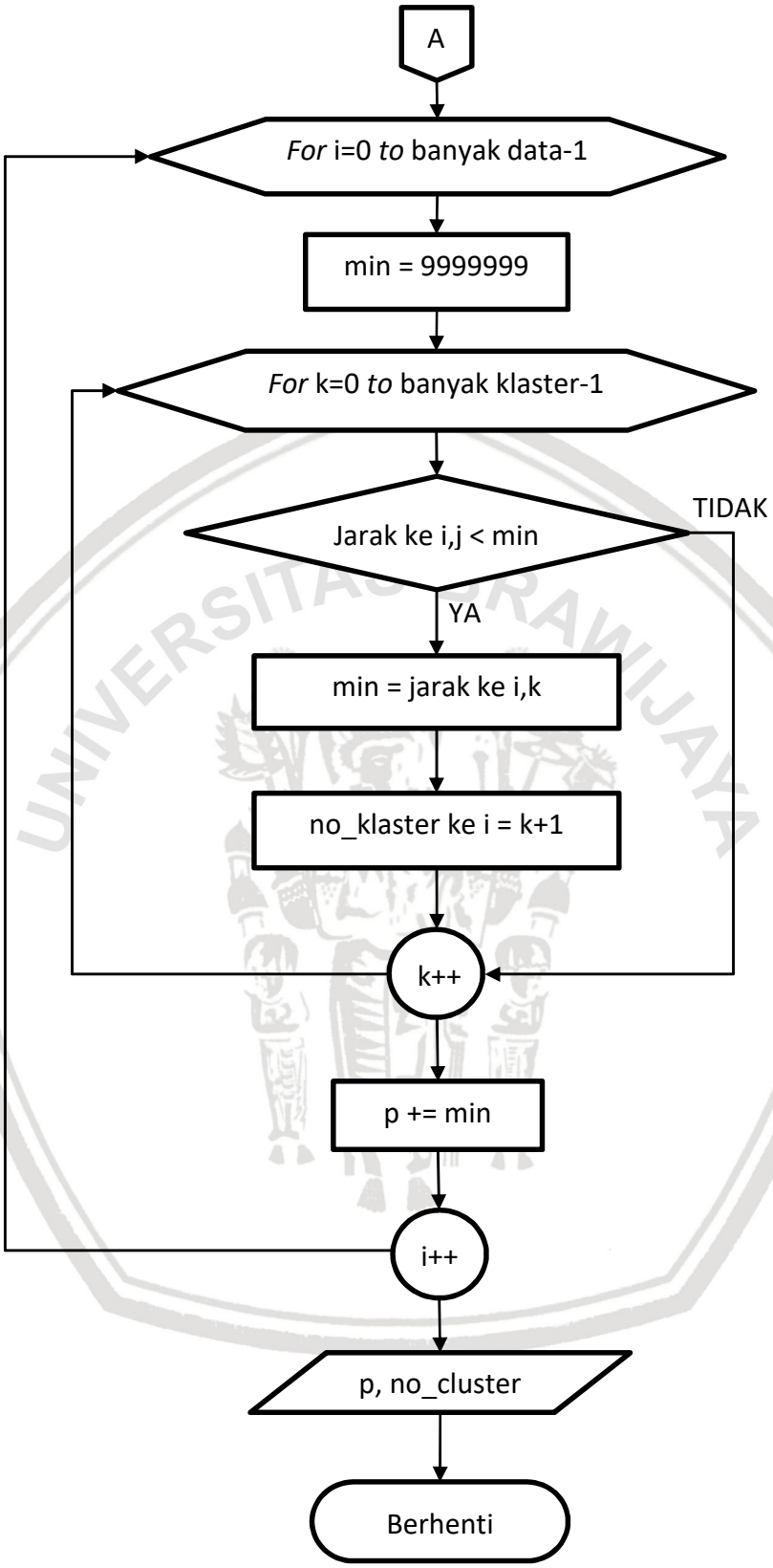
1. Membangun matriks data masukan dan matriks *centroid* hasil dari proses sebelumnya.
2. Menghitung jarak *Euclidean* antara data dengan pusat klaster menggunakan Persamaan 2.3.
3. Menyimpan hasil perhitungan ke dalam matriks jarak yang memiliki dimensi baris sejumlah data dan dimensi kolom sejumlah klaster.

4.2.2.4 Proses Hitung Objektif

Salah satu kriteria berhenti pada metode *K-Means* adalah ketika perubahan nilai fungsi objektif lebih kecil dari *threshold*, jika perubahan nilai fungsi objektif mencapai nilai 0 maka menandakan sudah tidak ada data yang berpindah klaster. Fungsi objektif didapat dari jumlah nilai jarak minimum setiap data ke semua klaster. Diagram alir proses hitung objektif ditampilkan pada Gambar 4.12 dan Gambar 4.13.



Gambar 4.12 Diagram Alir Hitung Objektif (Bagian 1)



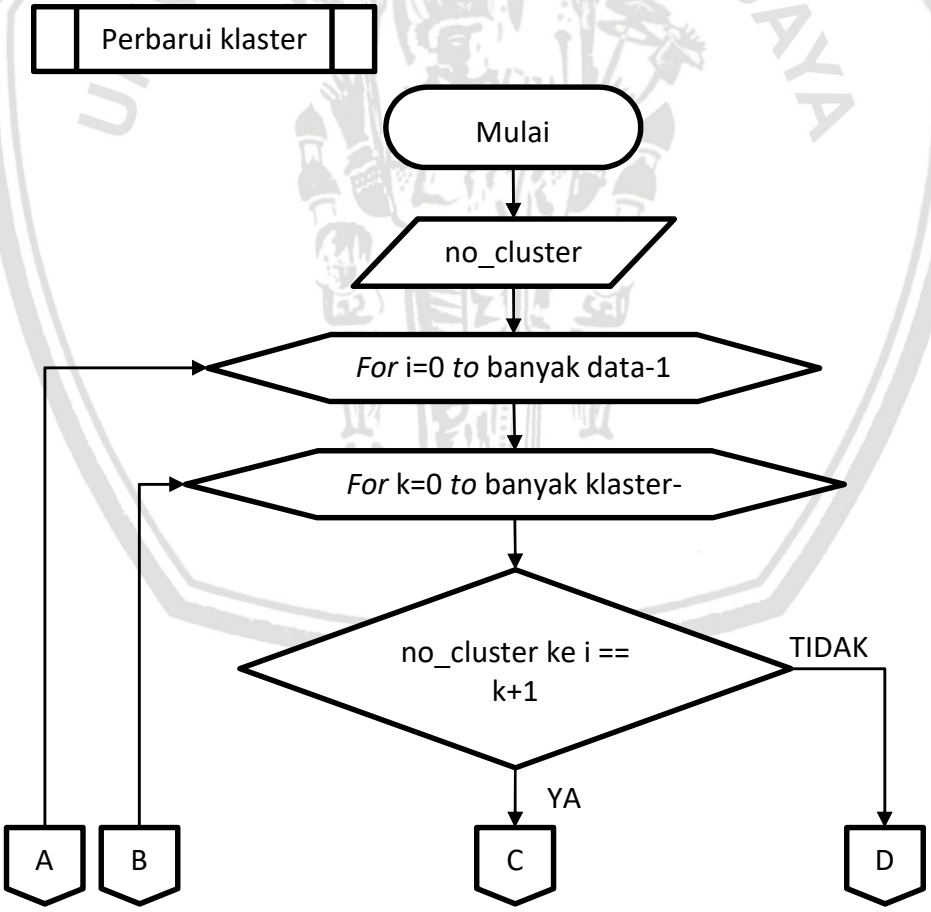
Gambar 4.13 Diagram Alir Hitung Objektif (Bagian 2)

Penjelasan dari diagram alir proses menghitung nilai fungsi objektif adalah sebagai berikut.

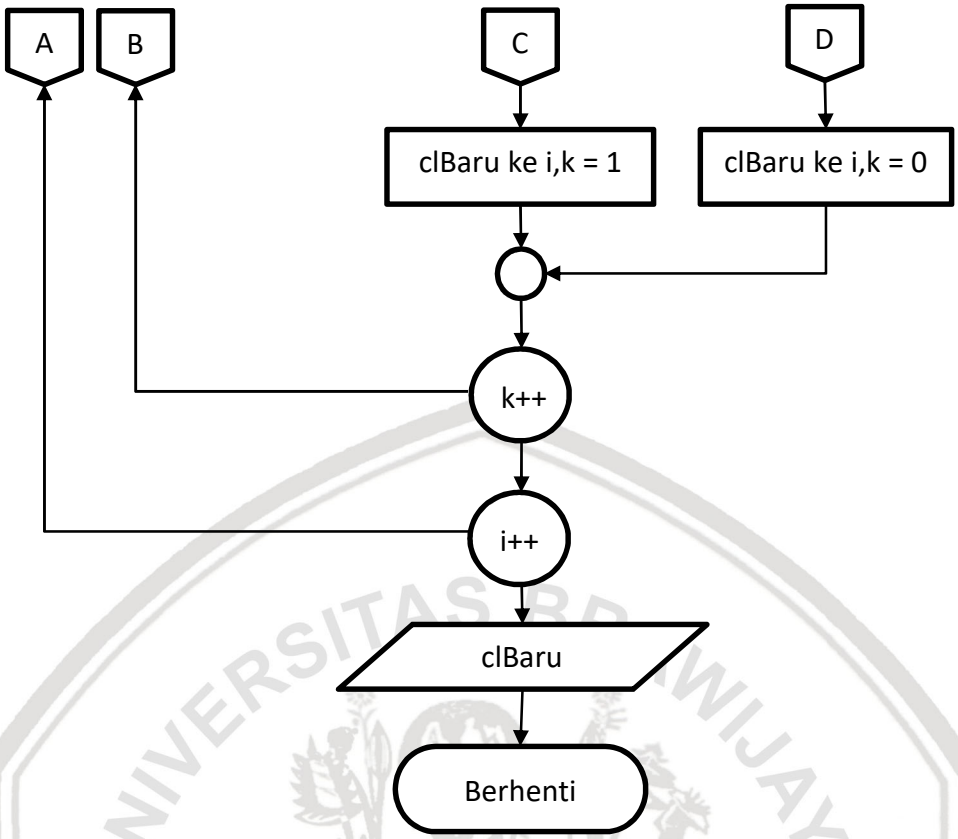
1. Membentuk matriks jarak yang didapatkan dari proses sebelumnya.
2. Mencari nilai minimum dari setiap baris pada matriks jarak.
3. Menyimpan indeks dari nilai jarak minimum ke daam matriks no_cluster.
4. Menjumlahkan nilai jarak minimum seluruh data dengan menggunkan Persamaan 2.4 dan menyimpannya ke dalam variabel p yang merupakan nilai fungsi objektif.

4.2.2.5 Proses Perbarui Klaster

Setiap iterasi, nilai *centroid* akan berubah dan memengaruhi keanggotaan klaster. Sehingga letak klaster untuk setiap data perlu diperbarui sesuai dengan jarak terdekat data tersebut dengan *centroid* klaster terbaru. Pada proses hitung objektif, telah disimpan indeks dari jarak minimum data dengan klaster, sehingga pada proses ini, akan merubah indeks menjadi bilangan biner sebagai penunjuk letak klaster seperti pada proses inisialisasi klaster awal. Diagram alir proses perbarui klaster ditampilkan pada Gambar 4.14 dan Gambar 4.15.



Gambar 4.14 Diagram Alir Proses Perbarui Klaster (Bagian 1)



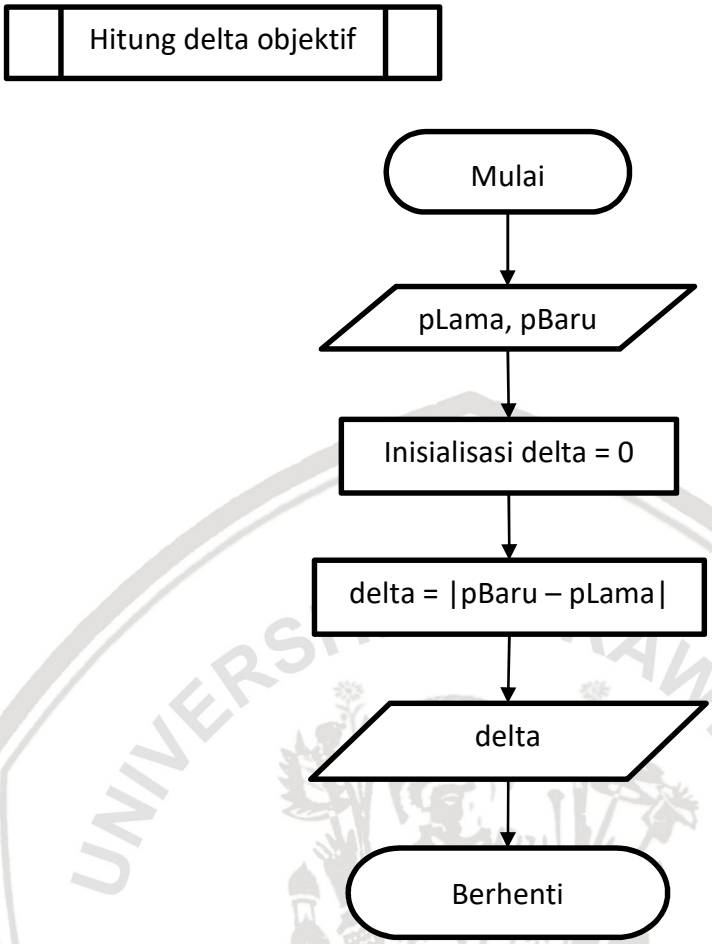
Gambar 4.15 Diagram Alir Perbarui Klaster (Bagian 2)

Penjelasan dari diagram alir proses memperbarui klaster adalah sebagai berikut.

1. Membentuk matriks no_cluster yang didapatkan dari proses sebelumnya. Matriks no_cluster menyimpan nomor klaster terdekat dari setiap data yang menandakan letak klaster dari data tersebut.
2. Melakukan perulangan sebanyak jumlah data, jika nilai pada matriks no_cluster indeks ke i sama dengan nilai variabel k, maka nilai pada matriks clBaru baris ke i kolom ke k akan diisi dengan 1 dan kolom lain diisi dengan 0. Matriks clBaru akan menyimpan nilai biner dari letak klaster setiap data dengan dimensi baris sejumlah data dan dimensi kolom sejumlah klaster.

4.2.2.6 Proses Hitung Perubahan Objektif

Untuk melakukan pengecekan kriteria berhenti, maka dilakukan proses perhitungan perubahan nilai fungsi objektif atau disebut delta objektif. Proses ini hanya mencari selisih absolut dari nilai fungsi objektif iterasi terbaru dengan nilai fungsi objektif pada iterasi sebelumnya. Diagram alir proses hitung perubahan objektif ditampilkan pada Gambar 4.16.



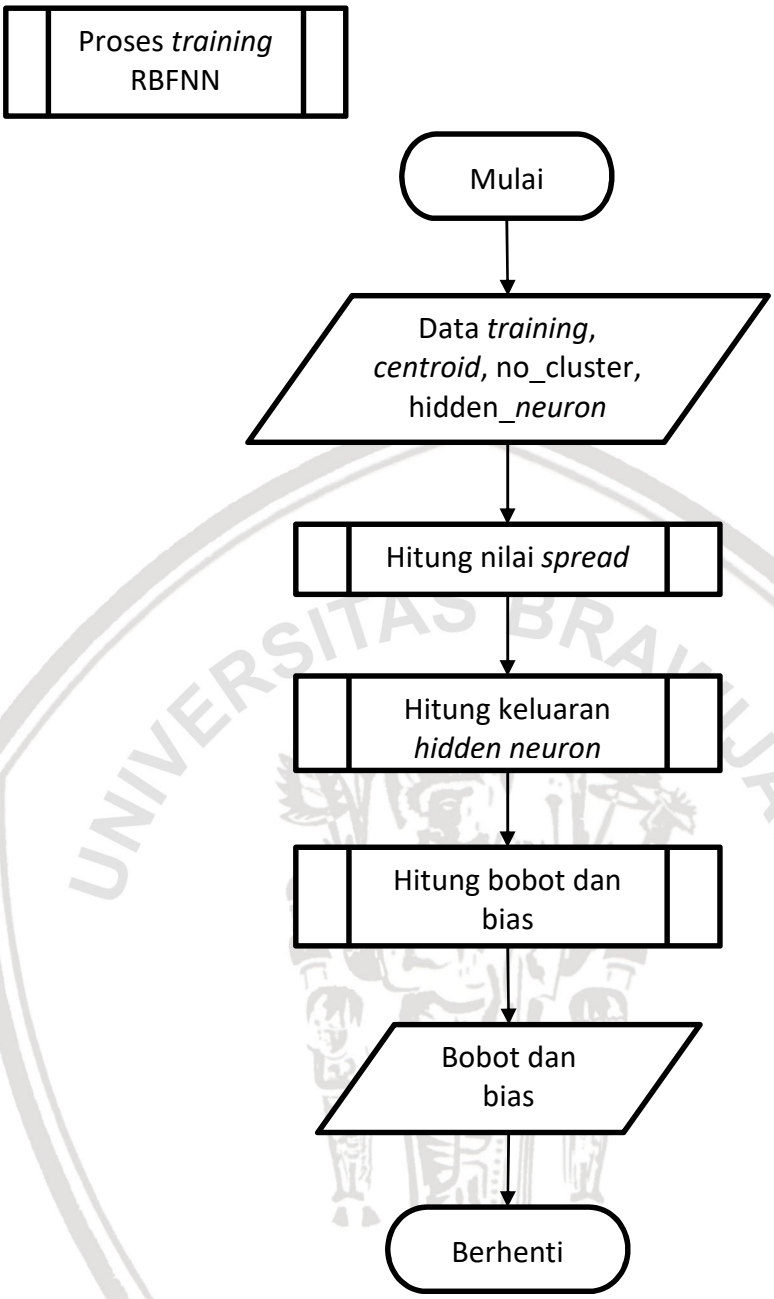
Gambar 4.16 Diagram Alir Hitung Delta Objektif

Penjelasan dari diagram alir proses menghitung delta atau perubahan nilai fungsi objektif adalah sebagai berikut.

1. Memasukkan nilai fungsi objektif iterasi sebelumnya ke dalam variabel pLama dan nilai fungsi objektif iterasi terbaru ke dalam variabel pBaru.
2. Menghitung perubahan nilai fungsi objektif dengan Persamaan 2.5.
3. Menyimpan nilai delta untuk dibandingkan dengan nilai *threshold* untuk melakukan pengecekan kriteria berhenti.

4.2.3 Proses *Training* RBFNN

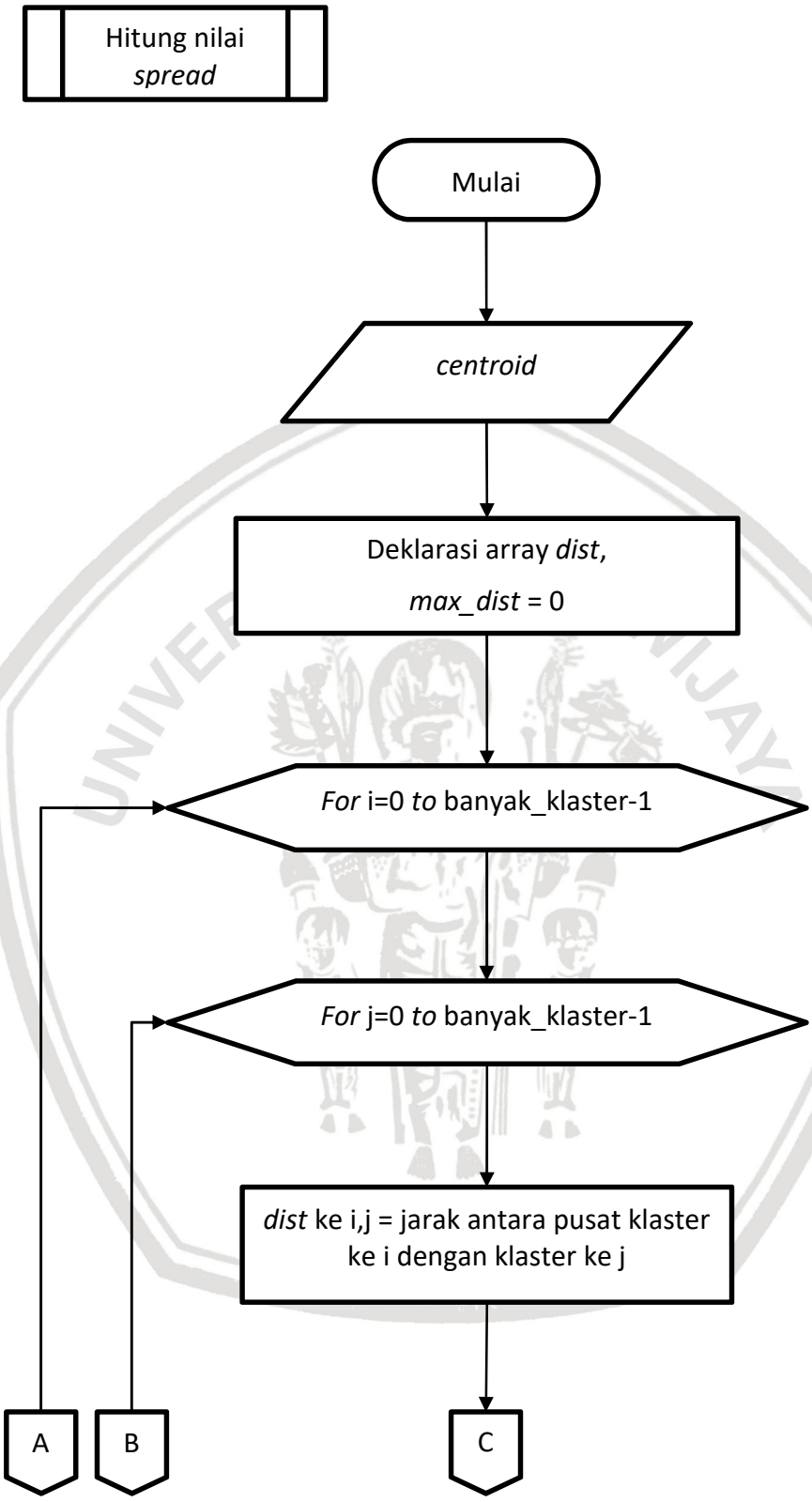
Proses *training* pada jaringan RBF terdiri dari beberapa sub proses. Proses training RBFNN memerlukan masukan berupa hasil *centroid* dan hasil *clustering* dari proses *K-Means*, data *training* dan jumlah hidden *neuron* yang sama dengan jumlah klaster. Kemudian *output* yang didapatkan dari proses *training* RBFNN adalah matriks bobot dan bias yang digunakan untuk proses *testing*. Diagram alir proses *training* RBFNN ditampilkan pada Gambar 4.17.



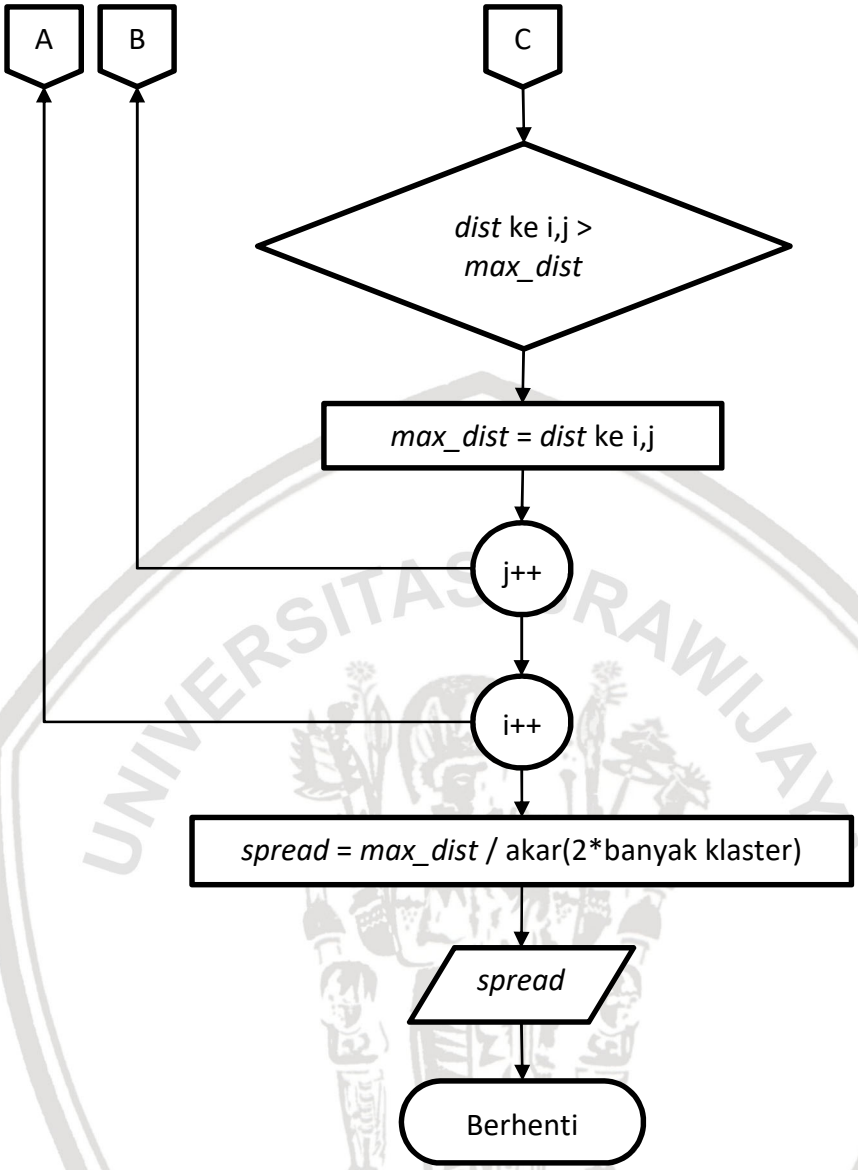
Gambar 4.17 Diagram Alir Proses *Training* RBFNN

4.2.3.1 Proses Hitung Nilai *Spread*

Salah satu parameter yang digunakan untuk fungsi aktivasi *Gaussian* adalah nilai *spread*. Nilai *spread* untuk semua *hidden neuron* diberikan nilai yang sama. Diagram alir proses menghitung nilai *spread* setiap *hidden neuron* ditampilkan pada Gambar 4.18 dan Gambar 4.19.



Gambar 4.18 Diagram Alir Hitung Nilai *Spread* (Bagian 1)



Gambar 4.19 Diagram Alir Hitung Nilai *Spread* (Bagian 2)

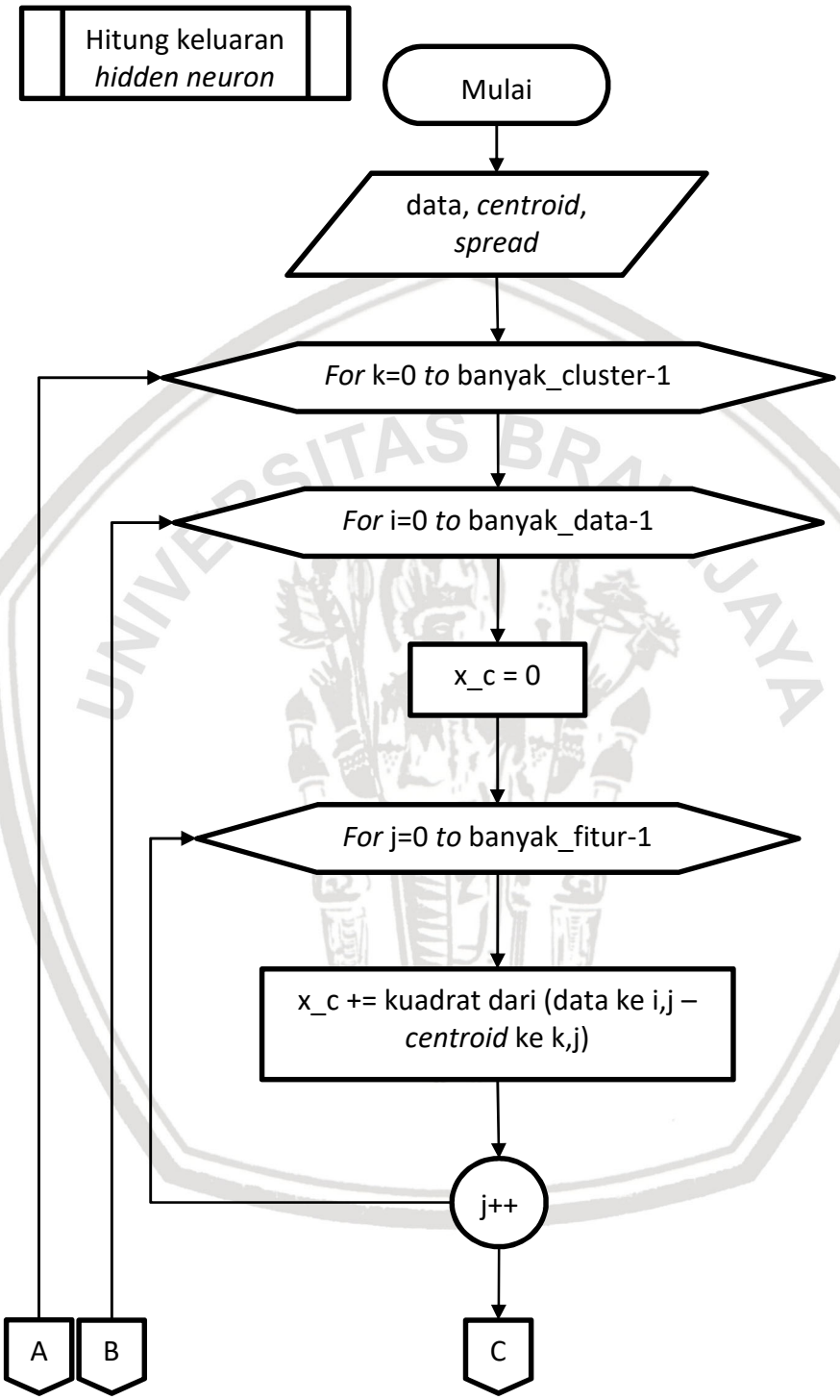
Penjelasan dari diagram alir proses menghitung nilai *spread* atau diameter klaster adalah sebagai berikut.

1. Memasukkan matriks *centroid* dari proses *K-Means*.
2. Hitung jarak *Euclidean* antar *centroid* klaster.
3. Cari jarak maksimum antara dua klaster.
4. Hitung nilai *spread* menggunakan Persamaan 2.6.
5. Menyimpan nilai *spread* untuk proses berikutnya.

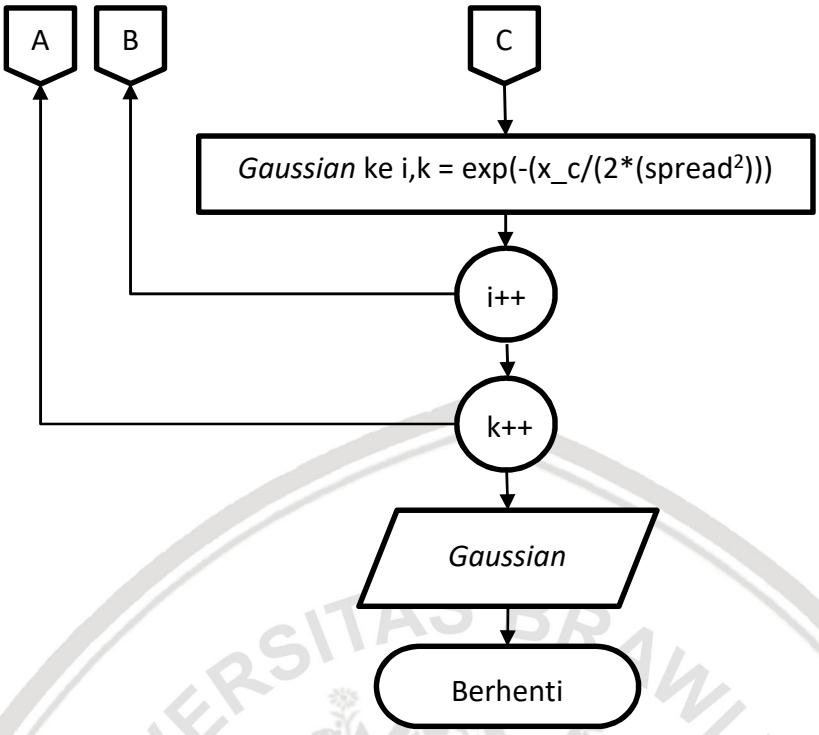
4.2.3.2 Proses Hitung Keluaran *Hidden Layer*

Pada jaringan RBF, pada *hidden layer* fungsi aktivasi yang digunakan bersifat *non-linear* yaitu dengan fungsi aktivasi *Gaussian*. Pada proses ini, masukan yang dibutuhkan diantaranya data, *centroid* dan *spread*. Keluaran yang dihasilkan

berupa matriks *Gaussian* yang akan digunakan untuk proses berikutnya. Diagram alir proses menghitung keluaran *hidden neuron* ditampilkan pada Gambar 4.20 dan 4.21.



Gambar 4.20 Diagram Alir Hitung Keluaran *Hidden Layer* (Bagian 1)



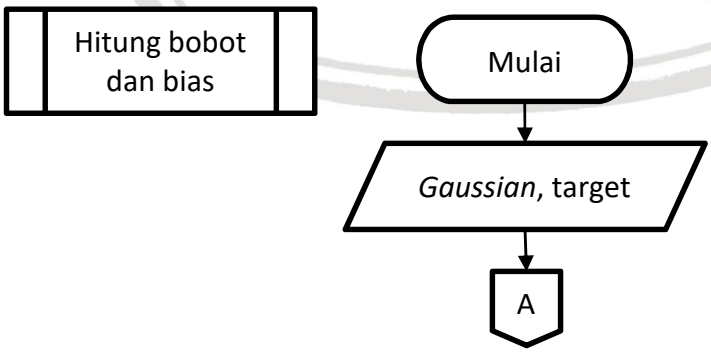
Gambar 4.21 Diagram Alir Hitung Keluaran *Hidden Layer* (Bagian 2)

Penjelasan dari diagram alir proses membentuk matriks *Gaussian* atau keluaran dari *hidden layer* adalah sebagai berikut.

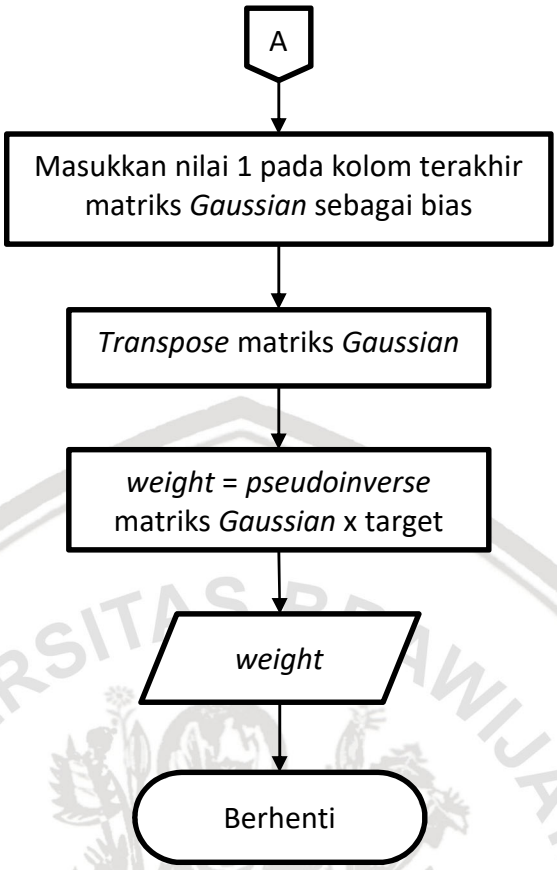
1. Untuk setiap *hidden neuron* menerima masukan berupa matriks data, matriks *centroid* dan nilai *spread*.
2. Kemudian menghitung jarak data dengan *centroid hidden neuron*.
3. Menghitung fungsi aktivasi *Gaussian* menggunakan Persamaan 2.7 untuk setiap *hidden neuron* dan menyimpan hasilnya menjadi matriks *Gaussian*.

4.2.3.3 Proses Hitung Bobot dan Bias

Hasil matriks *Gaussian* dari proses sebelumnya, dijadikan sebagai masukan untuk membentuk matriks bobot dan bias. Diagram alir proses menghitung bobot dan bias pada RBFNN ditampilkan pada Gambar 4.22 dan Gambar 4.23.



Gambar 4.22 Diagram Alir Hitung Bobot dan Bias (Bagian 1)



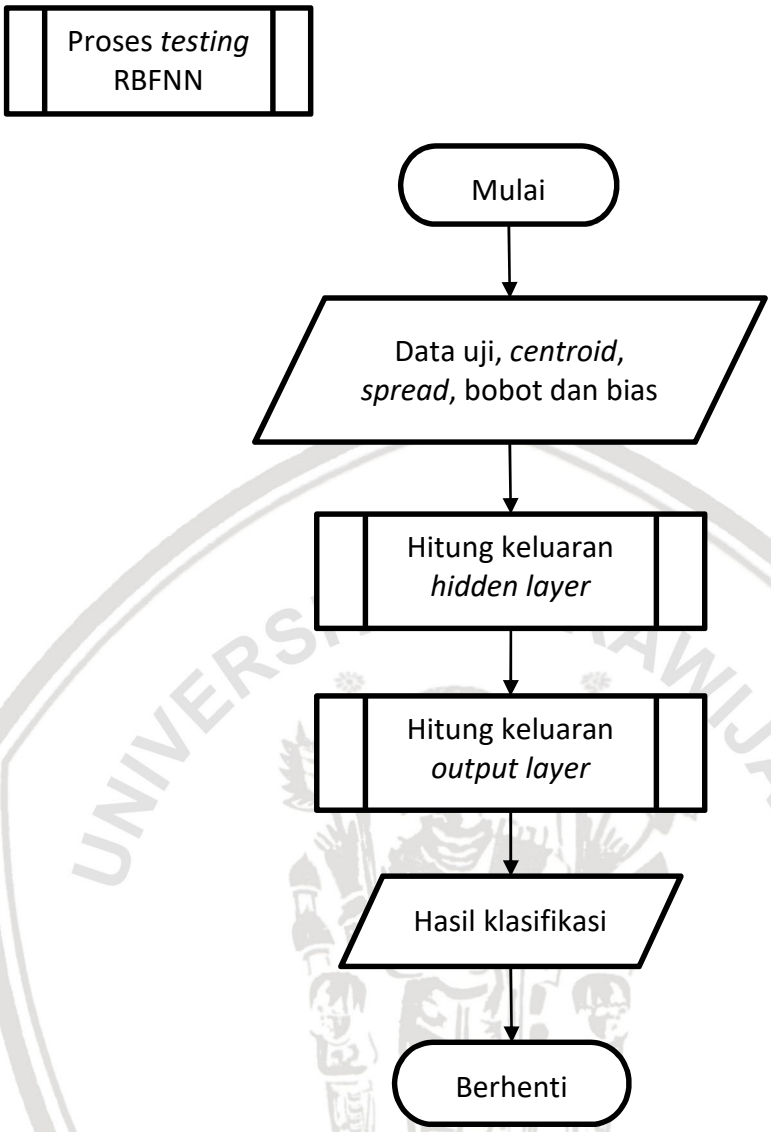
Gambar 4.23 Diagram Alir Hitung Bobot dan Bias (Bagian 2)

Penjelasan mengenai diagram alir proses untuk menghitung bobot dan bias adalah sebagai berikut.

1. Mengambil matriks *Gaussian* dari proses sebelumnya dan target dari data *training*.
2. Menambahkan satu kolom terakhir pada matriks *Gaussian* dan mengisinya dengan nilai 1 sebagai bias awal seperti pada Persamaan 2.8.
3. Melakukan *pseudoinverse* untuk matriks *Gaussian* yang baru dan mengalikan hasilnya dengan target dari data *training* menggunakan Persamaan 2.9.
4. Hasil dari perkalian tersebut disimpan dalam matriks *weight* yang terdiri dari jumlah baris sejumlah *hidden neuron* ditambah satu baris terakhir sebagai bias dan jumlah kolom sejumlah *output neuron* atau kelas.

4.2.4 Proses *Testing* RBFNN

Pada proses *testing*, masukan yang dibutuhkan didapatkan dari proses *training* yaitu berupa *centroid*, *spread*, bobot dan bias. Dari proses *testing* ini, setiap data akan diklasifikasikan ke dalam kelas-kelas sesuai dengan hasil dari *output layer*. Diagram alir proses *testing* RBFNN ditampilkan pada Gambar 4.24.



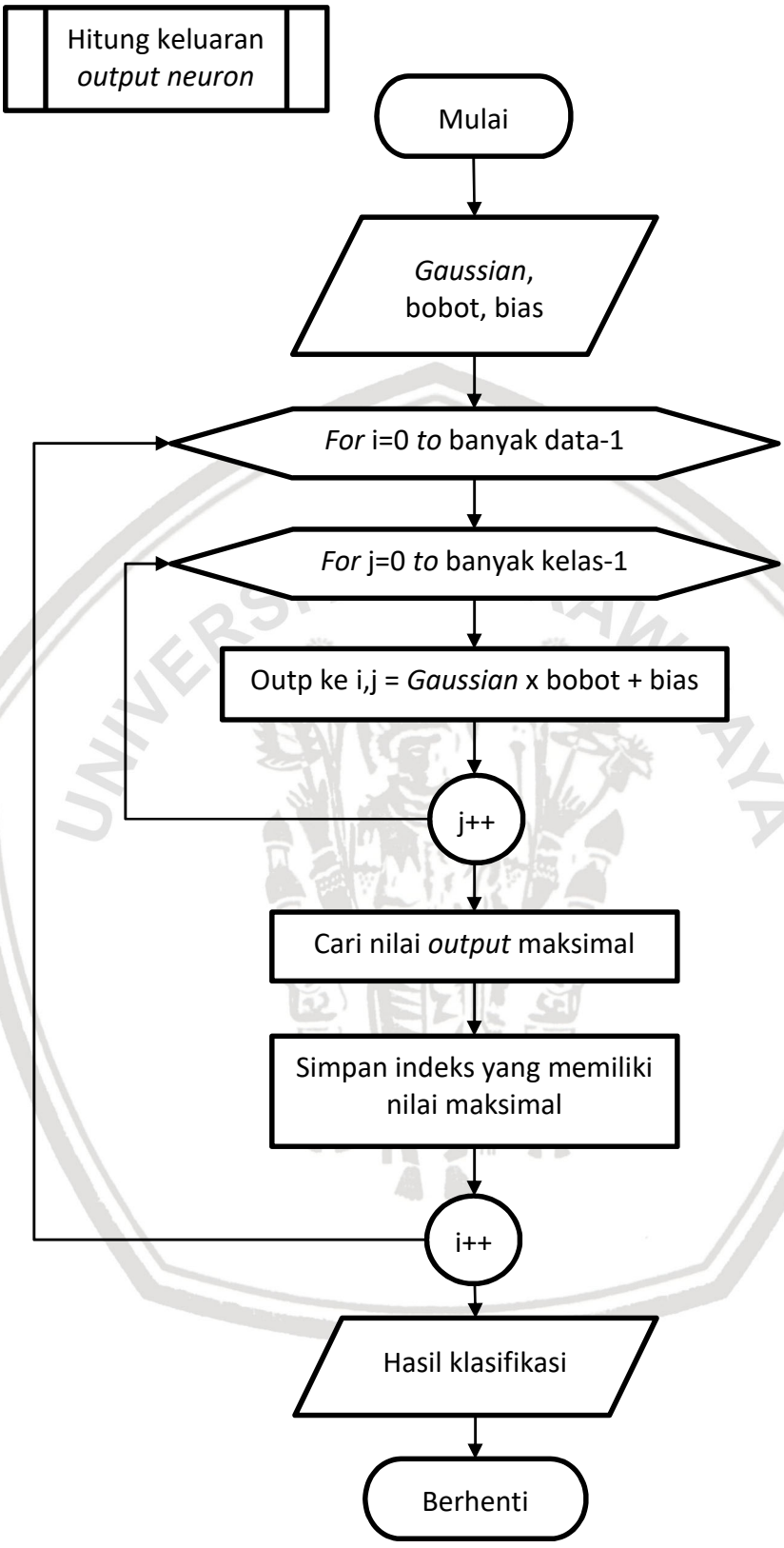
Gambar 4.24 Diagram Alir Proses *Testing* RBFNN

4.2.4.1 Hitung Keluaran *Hidden Layer*

Pada proses *testing* juga melakukan tahapan membentuk matriks *Gaussian*. Tahap ini, diagram alir sama seperti yang dilakukan pada proses *training* yaitu pada Gambar 2.20 dan Gambar 2.21. Hanya saja, data masukan yang digunakan adalah data uji yang berbeda dari data latih.

4.2.4.2 Hitung Keluaran *Output Layer*

Tahap akhir dari proses *testing* RBFNN adalah menghitung nilai keluaran dari *output layer*. Karena pada penelitian ini terdapat tujuh kelas, maka jumlah *output neuron* sebanyak tujuh *neuron*. Untuk *output layer* masukan yang dibutuhkan adalah matriks *Gaussian* dari data uji, bobot dan bias dari hasil *training*. Diagram alir proses menghitung keluaran *output layer* ditampilkan pada Gambar 4.25.



Gambar 4.25 Diagram Alir Hitung Keluaran *Output Layer*

Penjelasan mengenai diagram alir proses untuk menghitung keluaran *output layer* adalah sebagai berikut.

1. Menggunakan masukan berupa matriks *Gaussian* untuk data uji serta matriks bobot dan bias dari hasil *training*.
2. Menghitung hasil keluaran *output neuron* menggunakan Persamaan 2.10.
3. Menyimpan hasil klasifikasi data.

4.3 Perhitungan Manual RBFNN

4.3.1 Perhitungan Manual *K-Means*

1. Membentuk matriks data yang berisi data *training* seperti pada Tabel 4.1.

Tabel 4.1 Data Latih untuk Manualisasi

ID	X1	X2	X3	X4	X5	X6	Kelas
1	3	2	40	41	52	5	CL0
2	3	6	39	48	49	2	CL0
3	2	7	33	34	45	2	CL0
4	4	8	48	49	49	1	CL1
5	4	8	53	45	44	4	CL1
6	4	8	54	44	50	2	CL1
7	3	5	38	43	27	9	CL2
8	3	8	43	48	40	7	CL2
9	3	8	45	34	44	7	CL2
10	2	7	47	44	47	9	CL3
11	2	8	40	39	37	5	CL3
12	2	7	43	43	43	8	CL3
13	3	4	32	49	45	6	CL4
14	3	6	46	40	38	4	CL4
15	4	7	51	37	43	4	CL4
16	1	6	44	30	31	7	CL5
17	3	3	52	35	34	5	CL5
18	1	5	49	41	37	7	CL5
19	2	6	50	48	42	9	CL6
20	3	6	42	47	41	4	CL6
21	3	7	45	41	42	6	CL6

Dengan X1 = usia, X2 = pendidikan terakhir, X3 = Oscore, X4 = Ascore, X5 = Cscore dan X6 = BIS-11 atau *impulsivity*.

2. Normalisasi data latih dengan menggunakan Persamaan 2.1.

   Nilai hasil normalisasi data ke 1 fitur X1:

$$x'_{11} = \frac{3-1}{4-1} = 0{,}667$$

Nilai hasil normalisasi data ke 1 fitur X2:

$$x'_{12} = \frac{2-2}{8-2} = 0$$

Tabel 4.2 menampilkan data latih yang telah dilakukan normalisasi.

Tabel 4.2 Data Latih Hasil Normalisasi

ID	X1	X2	X3	X4	X5	X6
1	0,667	0	0,364	0,579	1	0,5
2	0,667	0,667	0,318	0,947	0,88	0,125
3	0,333	0,833	0,045	0,211	0,72	0,125
4	1	1	0,727	1	0,88	0
5	1	1	0,955	0,789	0,68	0,375
6	1	1	1	0,737	0,92	0,125
7	0,667	0,5	0,273	0,684	0	1
8	0,667	1	0,5	0,947	0,52	0,75
9	0,667	1	0,591	0,211	0,68	0,75
10	0,333	0,833	0,682	0,737	0,8	1
11	0,333	1	0,364	0,474	0,4	0,5
12	0,333	0,833	0,5	0,684	0,64	0,875
13	0,667	0,333	0	1	0,72	0,625
14	0,667	0,667	0,636	0,526	0,44	0,375
15	1	0,833	0,864	0,368	0,64	0,375
16	0	0,667	0,545	0	0,16	0,75
17	0,667	0,167	0,909	0,263	0,28	0,5
18	0	0,5	0,773	0,579	0,4	0,75
19	0,333	0,667	0,818	0,947	0,6	1
20	0,667	0,667	0,455	0,895	0,56	0,375
21	0,667	0,833	0,591	0,579	0,6	0,625

3. Inisialisasi klaster awal setiap data secara acak, misal didapatkan hasil acak seperti pada Tabel 4.3.

Tabel 4.3 Inisialisasi Klaster

ID	Klaster
1	1
2	1
3	1
4	2
5	2
6	2
7	3
8	3
9	3

Tabel 4.3 Inisialisasi Klaster (lanjutan)

10	4
11	4
12	4
13	5
14	5
15	5
16	6
17	6
18	6
19	7
20	7
21	7

4. Menghitung *centroid* setiap klaster dengan Persamaan 2.2.

Tabel 4.4 menampilkan jumlah anggota setiap klaster.

Tabel 4.4 Jumlah Anggota Setiap Klaster

Klaster	Jumlah
1	3
2	3
3	3
4	3
5	3
6	3
7	3

Jumlah fitur X1 dalam klaster 1:

$$Sum_{11} = 0{,}667 + 0{,}667 + 0{,}333 = 1{,}667$$

Jumlah fitur X2 dalam klaster 1:

$$Sum_{12} = 0 + 0{,}667 + 0{,}833 = 1{,}5$$

Tabel 4.5 menampilkan jumlah setiap fitur data pada setiap klaster.

Tabel 4.5 Jumlah Setiap Fitur Data

Klaster	X1	X2	X3	X4	X5	X6
1	1,667	1,5	0,727	1,737	2,6	0,75
2	3	3	2,682	2,526	2,48	0,5
3	2	2,5	1,364	1,842	1,2	2,5
4	1	2,667	1,545	1,895	1,84	2,375
5	2,333	1,833	1,5	1,895	1,8	1,375
6	0,667	1,333	2,227	0,842	0,84	2
7	1,667	2,167	1,864	2,421	1,76	2

Nilai *centroid* pada klaster ke 1 fitur X1:

$$C_{11} = \frac{1{,}667}{3} = 0{,}556$$

Nilai *centroid* pada klaster ke 1 fitur X2:

$$C_{12} = \frac{1{,}5}{3} = 0{,}5$$

Tabel 4.6 menampilkan nilai *centroid* atau pusat klaster setiap klaster.

Tabel 4.6 Pusat Klaster Iterasi Pertama

Klaster	X1	X2	X3	X4	X5	X6
1	0,556	0,5	0,242	0,579	0,867	0,25
2	1	1	0,894	0,842	0,827	0,167
3	0,667	0,833	0,455	0,614	0,4	0,833
4	0,333	0,889	0,515	0,632	0,613	0,792
5	0,778	0,611	0,5	0,632	0,6	0,458
6	0,222	0,444	0,742	0,281	0,28	0,667
7	0,556	0,722	0,621	0,807	0,587	0,667

5. Hitung jarak *Euclidean* data dengan pusat klaster menggunakan Persamaan 2.3.

Jarak data ke – 1 dengan *centroid* klaster 1

$$d_{11} = \sqrt{(0{,}667 - 0{,}556)^2 + (0 - 0{,}5)^2 + (0{,}364 - 0{,}242)^2 + (0{,}579 - 0{,}579)^2 + (1 - 0{,}867)^2 + (0{,}5 - 0{,}25)^2} = 0{,}598$$

Jarak data ke – 1 dengan *centroid* klaster 2

$$d_{12} = \sqrt{(0{,}667 - 0{,}556)^2 + (0{,}667 - 0{,}5)^2 + (0{,}318 - 0{,}242)^2 + (0{,}947 - 0{,}579)^2 + (0{,}88 - 0{,}867)^2 + (0{,}125 - 0{,}25)^2} = 1{,}266$$

Tabel 4.7 menampilkan jarak setiap data latih dengan pusat klaster.

Tabel 4.7 Matriks Jarak Data Latih dengan Pusat Klaster

ID	C1	C2	C3	C4	C5	C6	C7
1	0,598	1,266	1,084	1,078	0,754	1,083	0,922
2	0,444	0,755	0,943	0,903	0,581	1,235	0,712
3	0,61	1,267	1,022	0,926	0,87	1,068	1,03
4	0,961	0,289	1,134	1,156	0,822	1,495	0,924
5	1,024	0,267	0,842	0,923	0,667	1,209	0,693
6	1,032	0,181	1,106	1,114	0,822	1,375	0,91
7	1,157	1,466	0,58	0,862	0,856	0,88	0,808
8	0,914	0,844	0,403	0,484	0,596	1,036	0,368
9	0,897	0,982	0,543	0,559	0,664	0,837	0,68
10	0,972	1,106	0,605	0,347	0,79	0,87	0,476
11	0,778	1,075	0,527	0,437	0,659	0,737	0,605

Tabel 4.7 Matriks Jarak Data Latih dengan Pusat Klaster (lanjutan)

12	0,825	1,09	0, 421	0,117	0,652	0,747	0,371
13	0,662	1,265	0,867	0,927	0,719	1,214	0,779
14	0,629	0,762	0,529	0,624	0,279	0,655	0,448
15	0,898	0,576	0,779	0,901	0,556	0,997	0,741
16	1,23	1,651	0,96	0,876	1,135	0,488	1,076
17	1,036	1,245	0,949	1,058	0,786	0,575	0,906
18	1,028	1,362	0,815	0,615	0,905	0,405	0,689
19	1,085	1,148	0,671	0,533	0,834	0,852	0,471
20	0,543	0,73	0,585	0,64	0,309	0,933	0,37
21	0,676	0,749	0,321	0,388	0,317	0,752	0,282

Dengan C1 = klaster 1, C2 = klaster 2, C3 = klaster 3, C4 = klaster 4, C5 = klaster 5, C6 = klaster 6 dan C7 = klaster 7.

6. Mencari jarak minimum data dengan semua klaster untuk menentukan letak klaster terbaru dari data tersebut.

   Nilai minimal dari jarak data ke-1 dengan semua klaster.

$$Min_1 = \min[0{,}598; 1{,}266; 1{,}084; 1{,}078; 0{,}754; 1{,}083; 0{,}922] = 0{,}598$$

$$Argmin_1 = C1$$

   Nilai minimal dari jarak data ke-2 dengan semua klaster.

$$Min_1 = \min[0{,}444; 0{,}755; 0{,}943; 0{,}903; 0{,}581; 1{,}235; 0{,}712] = 0{,}444$$

$$Argmin_2 = C1$$

   Tabel 4.8 menampilkan nilai minimum jarak data ke semua klaster dan letak klaster baru dari data tersebut.

Tabel 4.8 Matriks Jarak Minimum dan Klaster Baru

ID	Min	Argmin	C-Lama
1	0,598	C1	C1
2	0,444	C1	C1
3	0,61	C1	C1
4	0,289	C2	C2
5	0,267	C2	C2
6	0,181	C2	C2
7	0,58	C3	C3
8	0,368	C7	C3

Tabel 4.8 Matriks Jarak Minimum dan Klaster Baru (lanjutan)

9	0,543	C3	C3
10	0,347	C4	C4
11	0,437	C4	C4
12	0,117	C4	C4
13	0,662	C1	C5
14	0,279	C5	C5
15	0,556	C5	C5
16	0,488	C6	C6
17	0,575	C6	C6
18	0,405	C6	C6
19	0,471	C7	C7
20	0,309	C7	C7
21	0,282	C7	C7

Dengan Argmin adalah letak klaster baru dari data tersebut dan C-Lama adalah letak klaster data pada iterasi sebelumnya.

7. Menghitung nilai fungsi objektif dengan menggunakan Persamaan 2.4.

$$F_0 = 0{,}598 + 0{,}444 + 0{,}61 + 0{,}289 + \cdots + 0{,}282 = 8{,}806$$

8. Menghitung perubahan nilai objektif dengan menggunakan Persamaan 2.5. Karena masih pada iterasi ke-0, maka perubahan sama dengan nilai F_0.

$$\Delta F_0 = F_0 = 8{,}806$$

9. Pada penelitian ini, nilai *threshold* yang digunakan adalah 0,00001 dan nilai maksimum iterasi yang digunakan adalah 100. Selanjutnya melakukan pengecekan mengenai kriteria berhenti. Jika salah satu kriteria berhenti terpenuhi maka iterasi dihentikan.

   Kriteria 1: $\Delta F_0 < threshold$

$$8{,}806 > 0{,}00001$$

   Belum terpenuhi, maka bernilai *false*.

   Kriteria 2: $iterasi = maksimum\ iterasi$

$$1 \neq 100$$

   Belum terpenuhi, maka bernilai *false*.

   Karena kedua kriteria berhenti masih bernilai *false*, maka iterasi dilanjutkan.

10. Ulangi langkah 4 – 9 dengan menggunakan letak klaster terbaru yang didapatkan dari langkah 6.

11. Setelah iterasi 3, salah satu kriteria berhenti terpenuhi, karena nilai $F_3 = 7{,}53$ dan $F_2 = 7{,}53$ sehingga $\Delta F_3 = 0$. Maka selanjutnya adalah menyimpan nilai *centroid* dan data hasil *clustering* pada iterasi terakhir. Tabel 4.9 menampilkan nilai *centroid* pada iterasi terakhir proses *K-Means*.

Tabel 4.9 Pusat Klaster Pada Iterasi Terakhir

Klaster	X1	X2	X3	X4	X5	X6
1	0,583	0,458	0,182	0,684	0,83	0,344
2	1	1	0,894	0,842	0,827	0,167
3	0,667	0,5	0,273	0,684	0	1
4	0,333	0,833	0,591	0,711	0,61	0,844
5	0,833	0,75	0,75	0,447	0,54	0,375
6	0,222	0,444	0,742	0,281	0,28	0,667
7	0,667	0,875	0,534	0,658	0,59	0,625

Tabel 4.10 menampilkan nomor klaster setiap data.

Tabel 4.10 Nomor Klaster Terbaru

ID	Nomor Klaster
1	1
2	1
3	1
4	2
5	2
6	2
7	3
8	7
9	4
10	7
11	3
12	4
13	1
14	5
15	5
16	6
17	6
18	5
19	7
20	7
21	4

4.3.2 Perhitungan Manual *Training* RBFNN

1. Menghitung nilai *spread* (σ).

   Pertama dengan menghitung jarak *Euclidean* antar *centroid* klaster.

Jarak *centroid* klaster ke 1 dengan ke 2 adalah sebagai berikut.

$$d_{12} = \sqrt{(0{,}583-1)^2+(0{,}458-1)^2+(0{,}182-0{,}894)^2+(0{,}684-0{,}842)^2+(0{,}83-0{,}827)^2+(0{,}344-0{,}167)^2} = 1{,}015$$

Tabel 4.11 menampilkan jarak *Euclidean* antar *centroid* klaster.

Tabel 4.11 Jarak *Euclidean* Antar *Centroid* Klaster

ID	C1	C2	C3	C4	C5	C6	C7
C1	0	1,015	1,066	0,818	0,782	1,007	0,665
C2	1,015	0	1,466	1,042	0,626	1,342	0,746
C3	1,066	1,466	0	0,849	1,028	0,88	0,836
C4	0,818	1,042	0,849	0	0,759	0,715	0,409
C5	0,782	0,626	1,028	0,759	0	0,805	0,446
C6	1,007	1,342	0,88	0,715	0,805	0	0,816
C7	0,665	0,746	0,836	0,409	0,446	0,816	0

Selanjutnya adalah mencari jarak maksimum antara 2 klaster. Dari matriks pada Tabel 4.11 di dapatkan jarak maksimum = 1,466. Kemudian nilai *spread* (σ) dihitung dengan menggunakan Persamaan 2.6.

$$\sigma = \frac{1{,}466}{\sqrt{2 \times 7}} = 0{,}392$$

Karena pada penelitian ini nilai *spread* yang digunakan untuk semua *hidden neuron* bernilai sama, maka matriks nilai *spread* yang dibangun adalah seperti yang ditampilkan pada Tabel 4.12.

Tabel 4.12 Nilai *Spread* Setiap *Hidden Neuron*

1	2	3	4	5	6	7
0,392	0,392	0,392	0,392	0,392	0,392	0,392

2. Menghitung keluaran *hidden layer* menggunakan Persamaan 2.7.

   Nilai gaussian untuk data ke-1 pada *hidden neuron* ke-1 adalah sebagai berikut.

$$\varphi_{11} = \exp - \left(\frac{(0{,}667-0{,}583)^2+(0-0{,}458)^2+(0{,}364-0{,}182)^2+(0{,}579-0{,}684)^2+(1-0{,}83)^2+(0{,}5-0{,}344)^2}{2\,(0{,}392)^2} \right) = 0{,}359$$

   Nilai gaussian untuk data ke-2 pada *hidden neuron* ke-1 adalah sebagai berikut.

$$\varphi_{21} = \exp - \left(\frac{(0{,}67 - 0{,}583)^2 + (0{,}67 - 0{,}458)^2 + (0{,}32 - 0{,}182)^2 + (0{,}95 - 0{,}684)^2 + (0{,}88 - 0{,}83)^2 + (0{,}13 - 0{,}344)^2}{2\,(0{,}392)^2} \right)$$
$$= 0{,}541$$

Tabel 4.13 menampilkan matriks *Gaussian* yang merupakan keluaran dari *hidden layer*.

Tabel 4.13 Matriks *Gaussian* Data Latih

ID	G1	G2	G3	G4	G5	G6	G7
1	0,359	0,005	0,007	0,024	0,041	0,022	0,041
2	0,541	0,157	0,005	0,061	0,121	0,007	0,191
3	0,192	0,005	0,003	0,03	0,053	0,024	0,07
4	0,04	0,762	0	0,012	0,12	0,001	0,085
5	0,028	0,793	0,004	0,066	0,417	0,009	0,28
6	0,02	0,899	0	0,017	0,236	0,002	0,099
7	0,025	0,001	1	0,096	0,032	0,081	0,103
8	0,092	0,099	0,101	0,488	0,17	0,03	0,674
9	0,057	0,043	0,028	0,27	0,339	0,102	0,454
10	0,056	0,019	0,035	0,798	0,073	0,085	0,346
11	0,123	0,023	0,068	0,379	0,198	0,17	0,456
12	0,132	0,021	0,103	0,965	0,126	0,162	0,557
13	0,448	0,005	0,061	0,062	0,023	0,008	0,098
14	0,243	0,151	0,082	0,252	0,812	0,247	0,601
15	0,051	0,34	0,008	0,061	0,812	0,039	0,3
16	0,005	0	0,028	0,061	0,018	0,461	0,026
17	0,026	0,006	0,036	0,029	0,19	0,341	0,052
18	0,033	0,002	0,049	0,347	0,048	0,587	0,102
19	0,031	0,014	0,06	0,594	0,052	0,094	0,224
20	0,453	0,177	0,072	0,26	0,35	0,059	0,576
21	0,225	0,161	0,095	0,563	0,627	0,159	0,964

3. Menghitung matriks bobot dan bias dengan melakukan *pseudoinverse* matriks *Gaussian* menggunakan Persamaan 2.9.

   Sebelum melakukan *pseudoinverse*, tambahkan satu kolom terakhir pada matriks *Gaussian* dan mengisinya dengan nilai 1 sebagai bias awal. Tabel 4.14 menampilkan matriks *Gaussian* terbaru untuk mencari bobot dan bias.

Tabel 4.14 Matriks *Gaussian* dan Bias

ID	G1	G2	G3	G4	G5	G6	G7	B
1	0,359	0,005	0,007	0,024	0,041	0,022	0,041	1
2	0,541	0,157	0,005	0,061	0,121	0,007	0,191	1
3	0,192	0,005	0,003	0,03	0,053	0,024	0,07	1
4	0,04	0,762	0	0,012	0,12	0,001	0,085	1
5	0,028	0,793	0,004	0,066	0,417	0,009	0,28	1

Tabel 4.14 Matriks *Gaussian* dan Bias (lanjutan)

6	0,02	0,899	0	0,017	0,236	0,002	0,099	1
7	0,025	0,001	1	0,096	0,032	0,081	0,103	1
8	0,092	0,099	0,101	0,488	0,17	0,03	0,674	1
9	0,057	0,043	0,028	0,27	0,339	0,102	0,454	1
10	0,056	0,019	0,035	0,798	0,073	0,085	0,346	1
11	0,123	0,023	0,068	0,379	0,198	0,17	0,456	1
12	0,132	0,021	0,103	0,965	0,126	0,162	0,557	1
13	0,448	0,005	0,061	0,062	0,023	0,008	0,098	1
14	0,243	0,151	0,082	0,252	0,812	0,247	0,601	1
15	0,051	0,34	0,008	0,061	0,812	0,039	0,3	1
16	0,005	0	0,028	0,061	0,018	0,461	0,026	1
17	0,026	0,006	0,036	0,029	0,19	0,341	0,052	1
18	0,033	0,002	0,049	0,347	0,048	0,587	0,102	1
19	0,031	0,014	0,06	0,594	0,052	0,094	0,224	1
20	0,453	0,177	0,072	0,26	0,35	0,059	0,576	1
21	0,225	0,161	0,095	0,563	0,627	0,159	0,964	1

Dengan G1 sampai G7 merupakan keluaran *hidden neuron* ke-1 sampai ke-7.

Tabel 4.15 menampilkan matriks target yang berisi kelas aktual dari data *training* yang telah dikonversi menjadi nilai biner.

Tabel 4.15 Matriks Target

ID	CL0	CL1	CL2	CL3	CL4	CL5	CL6
1	1	0	0	0	0	0	0
2	1	0	0	0	0	0	0
3	1	0	0	0	0	0	0
4	0	1	0	0	0	0	0
5	0	1	0	0	0	0	0
6	0	1	0	0	0	0	0
7	0	0	1	0	0	0	0
8	0	0	1	0	0	0	0
9	0	0	1	0	0	0	0
10	0	0	0	1	0	0	0
11	0	0	0	1	0	0	0
12	0	0	0	1	0	0	0
13	0	0	0	0	1	0	0
14	0	0	0	0	1	0	0
15	0	0	0	0	1	0	0
16	0	0	0	0	0	1	0
17	0	0	0	0	0	1	0
18	0	0	0	0	0	1	0
19	0	0	0	0	0	0	1
20	0	0	0	0	0	0	1
21	0	0	0	0	0	0	1

Hasil dari perkalian *pseudoinverse* matriks *Gaussian* dengan target data *training* ditampilkan pada Tabel 4.16.

Tabel 4.16 Matriks Bobot dan Bias

	O1	O2	O3	O4	O5	O6	O7
W1	0,706	-0,176	-1,414	0,027	0,653	-0,261	0,465
W2	-0,45	1,287	-0,57	0,004	-0,352	-0,035	0,116
W3	-0,387	-0,009	0,538	-0,059	0,08	-0,234	0,07
W4	-0,278	-0,048	-1,001	1,169	0,418	-0,402	0,141
W5	-0,125	-0,407	-0,755	0,06	1,645	-0,309	-0,109
W6	-0,736	0,079	-0,858	-0,242	-0,282	1,905	0,134
W7	-0,353	0,078	1,386	-0,486	-1,224	0,012	0,588
BIAS	0,45	0,017	0,538	0,003	0,013	0,137	-0,159

Dengan W1 sampai W7 adalah bobot dari *hidden neuron* ke-1 sampai ke-7 dan O1 sampai O7 adalah bobot menuju *output neuron* ke-1 sampai ke-7.

4.3.3 Perhitungan Manual *Testing* RBFNN

1. Membentuk matriks data yang berisi data uji seperti pada Tabel 4.17.

Tabel 4.17 Data Uji untuk Manualisasi

ID	X1	X2	X3	X4	X5	X6	Kelas
1	3	6	42	37	42	4	CL0
2	3	6	32	50	37	4	CL0
3	4	2	34	42	48	2	CL1
4	4	4	55	34	42	3	CL1
5	2	7	45	53	48	2	CL2
6	4	2	44	44	41	7	CL2
7	3	5	34	45	47	4	CL3
8	3	8	45	35	42	3	CL3
9	1	2	41	38	34	8	CL4
10	3	8	47	36	49	4	CL4
11	2	7	43	33	43	3	CL5
12	3	5	48	49	45	3	CL5
13	3	4	54	36	44	5	CL6
14	2	2	47	48	39	4	CL6

2. Melakukan normalisasi untuk data uji dengan Persamaan 2.1, sehingga didapatkan nilai seperti yang ditampilkan pada Tabel 4.18.

Tabel 4.18 Hasil Normalisasi Data Uji untuk Manualisasi

ID	X1	X2	X3	X4	X5	X6	Kelas
1	0,667	0,667	0,435	0,2	0,533	0,333	CL0
2	0,667	0,667	0	0,85	0,2	0,333	CL0
3	1	0	0,087	0,45	0,933	0	CL1
4	1	0,333	1	0,05	0,533	0,167	CL1
5	0,333	0,833	0,565	1	0,933	0	CL2
6	1	0	0,522	0,55	0,467	0,833	CL2
7	0,667	0,5	0,087	0,6	0,867	0,333	CL3
8	0,667	1	0,565	0,1	0,533	0,167	CL3
9	0	0	0,391	0,25	0	1	CL4
10	0,667	1	0,652	0,15	1	0,333	CL4
11	0,333	0,833	0,478	0	0,6	0,167	CL5
12	0,667	0,5	0,696	0,8	0,733	0,167	CL5
13	0,667	0,333	0,957	0,15	0,667	0,5	CL6
14	0,333	0	0,652	0,75	0,333	0,333	CL6

3. Menghitung keluaran *hidden layer* dengan menggunakan *centroid*, *spread*, bobot dan bias dari proses *training*. Persamaan yang digunakan sama seperti proses *training* yaitu Persamaan 2.7.

Nilai gaussian untuk data ke-1 pada *hidden neuron* ke-1 adalah sebagai berikut.

$$\varphi_{11} = \exp - \left(\frac{(0{,}67-0{,}58)^2 + (0{,}67-0{,}46)^2 + (0{,}44-0{,}18)^2 + (0{,}2-0{,}68)^2 + (0{,}5-0{,}83)^2 + (0{,}33-0{,}34)^2}{2\,(0{,}392)^2} \right) = 0{,}43$$

Nilai gaussian untuk data ke-2 pada *hidden neuron* ke-1 adalah sebagai berikut.

$$\varphi_{21} = \exp - \left(\frac{(0{,}67-0{,}58)^2 + (0{,}67-0{,}46)^2 + (0-0{,}18)^2 + (0{,}85-0{,}68)^2 + (0{,}2-0{,}83)^2 + (0{,}33-0{,}34)^2}{2\,(0{,}392)^2} \right) = 0{,}403$$

Tabel 4.19 menampilkan matriks *Gaussian* yang merupakan keluaran dari *hidden layer*.

Tabel 4.19 Matriks *Gaussian* Data Uji

ID	G1	G2	G3	G4	G5	G6	G7
1	0,43	0,287	0,276	0,346	0,568	0,397	0,469
2	0,403	0,216	0,368	0,291	0,303	0,242	0,371
3	0,38	0,177	0,143	0,146	0,23	0,155	0,196
4	0,223	0,252	0,161	0,184	0,419	0,27	0,247

Tabel 4.19 Matriks *Gaussian* Data Uji (lanjutan)

5	0,379	0,359	0,146	0,298	0,3	0,201	0,334
6	0,3	0,181	0,347	0,249	0,305	0,282	0,289
7	0,813	0,262	0,242	0,316	0,354	0,249	0,404
8	0,296	0,307	0,194	0,283	0,504	0,301	0,391
9	0,172	0,078	0,299	0,215	0,157	0,385	0,173
10	0,311	0,342	0,156	0,304	0,445	0,253	0,391
11	0,31	0,22	0,182	0,283	0,379	0,353	0,329
12	0,482	0,441	0,218	0,34	0,498	0,293	0,438
13	0,287	0,251	0,212	0,292	0,463	0,425	0,331
14	0,332	0,182	0,268	0,273	0,287	0,387	0,27

4. Menghitung keluaran *output layer* dengan Persamaan 2.10. Bobot dan bias yang digunakan adalah hasil dari proses *training*.

   Nilai keluaran dari data ke-1 pada *output neuron* ke-1 adalah sebagai berikut.

   $$Y_{11} = (0{,}43 \times 0{,}706) + (0{,}287 \times -0{,}45) + (0{,}276 \times -0{,}387) + (0{,}346 \times -0{,}278) + (0{,}568 \times -0{,}125) + (0{,}397 \times -0{,}736) + (0{,}47 \times -0{,}353) + 0{,}45 = -0{,}108$$

   Nilai keluaran dari data ke-2 pada *output neuron* ke-1 adalah sebagai berikut.

   $$Y_{11} = (0{,}403 \times 0{,}706) + (0{,}216 \times -0{,}45) + (0{,}368 \times -0{,}387) + (0{,}291 \times -0{,}278) + (0{,}303 \times -0{,}125) + (0{,}242 \times -0{,}736) + (0{,}371 \times -0{,}353) + 0{,}45 = 0{,}0671$$

   Tabel 4.20 menampilkan hasil keluaran *output layer*.

Tabel 4.20 Matriks Hasil Keluaran *Output Layer*

ID	Y1	Y2	Y3	Y4	Y5	Y6	Y7
1	-0,108	0,129	-0,551	0,115	0,607	0,398	0,41
2	0,067	0,133	-0,17	0,113	0,327	0,192	0,338
3	0,33	0,104	-0,204	0,057	0,366	0,166	0,18
4	0,042	0,162	-0,225	0,056	0,471	0,347	0,147
5	0,113	0,318	-0,358	0,162	0,298	0,165	0,302
6	0,03	0,103	-0,123	0,092	0,345	0,317	0,235
7	0,355	0,102	-0,87	0,147	0,622	0,101	0,543
8	-0,056	0,195	-0,333	0,1	0,498	0,312	0,283
9	-0,003	0,055	-0,013	0,074	0,149	0,619	0,118
10	-0,009	0,257	-0,329	0,135	0,418	0,234	0,295
11	-0,003	0,131	-0,345	0,111	0,392	0,452	0,263
12	-0,02	0,337	-0,638	0,149	0,533	0,216	0,422
13	-0,112	0,145	-0,444	0,105	0,488	0,556	0,26
14	0,007	0,113	-0,339	0,108	0,333	0,524	0,254

Dengan Y1 sampai Y7 adalah *output neuron* ke-1 sampai ke-7.

5. Menentukan kelas dari setiap data dengan mencari nilai *output* maksimum dan *output neuron* yang memiliki nilai maksimum menandakan kelas dari data tersebut.

   Nilai Y maksimal data ke-1 adalah sebagai berikut.

   $Max_1 = \max[-0{,}108; 0{,}129; -0{,}551; 0{,}115; 0{,}607; 0{,}398; 0{,}41] = 0{,}607$

   $Argmax_1 = Y5$, sehingga data ke-1 masuk ke kelas CL4.

   Nilai Y maksimal data ke-2 adalah sebagai berikut.

   $Max_2 = \max[0{,}067; 0{,}133; -0{,}17; 0{,}113; 0{,}327; 0{,}192; 0{,}338] = 0{,}338$

   $Argmax_2 = Y7$, sehingga data ke-2 masuk ke kelas CL6.

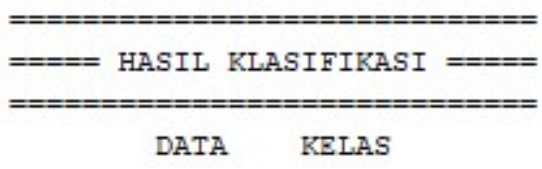
   Tabel 4.21 menampilkan hasil klasifikasi dari data uji.

Tabel 4.21 Hasil Klasifikasi Data Uji

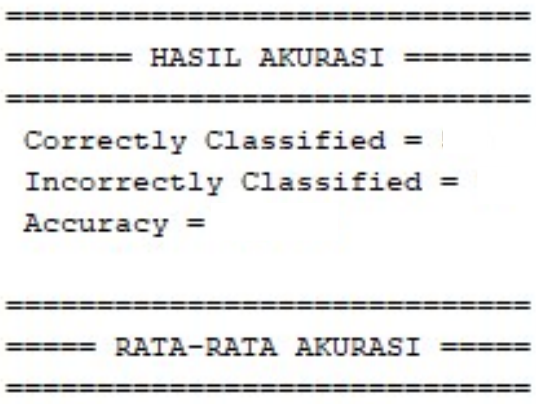
ID	Max	Argmax	Kelas
1	0,607	Y5	CL4
2	0,338	Y7	CL6
3	0,366	Y5	CL4
4	0,471	Y5	CL4
5	0,318	Y2	CL1
6	0,345	Y5	CL4
7	0,622	Y5	CL4
8	0,498	Y5	CL4
9	0,619	Y6	CL5
10	0,418	Y5	CL4
11	0,452	Y6	CL5
12	0,533	Y5	CL4
13	0,556	Y6	CL5
14	0,524	Y6	CL5

4.4 Perancangan Antarmuka

Pada perancangan antarmuka, terdapat dua hasil yang akan ditampilkan, yaitu hasil klasifikasi dan hasil akurasi sistem. Perancangan antarmuka untuk penelitian ini ditampilkan pada Gambar 4.26 dan Gambar 4.27.



Gambar 4.26 Perancangan Antarmuka Hasil Klasifikasi



Gambar 4.27 Perancangan Antarmuka Hasil Akurasi

4.5 Pengujian Metode

4.5.1 Pengujian Maksimum Iterasi *K-Means*

Salah satu kriteria berhenti pada *K-Means* adalah ketika iterasi telah mencapai nilai iterasi maksimum. Untuk mengetahui pengaruh nilai maksimum iterasi dengan hasil klasifikasi, maka dilakukan pengujian dengan beberapa variasi nilai maksimum iterasi. Perancangan mengenai pengujian maksimum iterasi *K-Means* ditampilkan pada Tabel 4.22.

Tabel 4.22 Perancangan Pengujian Maksimum Iterasi *K-Means*

Maksimum Iterasi *K-Means*	Akurasi
5	
10	
20	
50	
100	

4.5.2 Pengujian Banyak *Hidden Neuron*

Pada RBFNN, banyak *hidden neuron* merepresentasikan banyaknya klaster yang terbentuk. Untuk mengetahui pengaruh banyak *hidden neuron* atau banyak

klaster dengan hasil klasifikasi, maka dilakukan pengujian dengan beberapa variasi banyak *hidden neuron*. Perancangan mengenai pengujian banyak *hidden neuron* ditampilkan pada Tabel 4.23.

Tabel 4.23 Perancangan Pengujian Banyak *Hidden Neuron*

Banyak *Hidden Neuron*	Akurasi
2	
3	
4	
5	
6	
7	
8	
9	
10	
20	
50	

BAB 5 IMPLEMENTASI

5.1 Kode Sumber

5.1.1 Implementasi Normalisasi

Untuk meminimalkan rentang nilai antar fitur maka perlu dilakukan normalisasi. Fungsi normalisasi menerima masukan berupa *array* dua dimensi yang berisi nilai fitur data dan menghasilkan *array* dua dimensi yang berisi nilai normal dari setiap fitur. Implementasi kode sumber untuk normalisasi ditampilkan pada Algoritme 1.

Algoritme 1: Fungsi Normalisasi

```
1  public double[][] normalisasi(double[][] fitur) {
2     double[] max = new double[fitur[0].length];
3     double[] min = new double[fitur[0].length];
4     for (int i = 0; i < fitur[0].length; i++) {
5        double temp_min = Double.MAX_VALUE;
6        double temp_max = Double.MIN_VALUE;
7        for (int j = 0; j < fitur.length; j++) {
8           //Min
9           if (fitur[j][i] < temp_min) {
10             temp_min = fitur[j][i];
11          }
12          if (fitur[j][i] > temp_max) {
13             temp_max = fitur[j][i];
14          }
15       }
16       min[i] = temp_min;
17       max[i] = temp_max;
18    }
19    for (int i = 0; i < fitur.length; i++) {
20       for (int j = 0; j < fitur[0].length; j++) {
21          fitur[i][j] = (fitur[i][j] - min[j]) / (max[j] - min[j]);
22       }
23    }
24    return fitur;
25 }
```

Baris 4-18 berfungsi untuk menemukan nilai maksimal dan minimal dari setiap fitur, yang kemudian disimpan dalam *array* ‘min’ untuk nilai minimum dan ‘max’ untuk nilai maksimum.

Baris 19-23 berfungsi untuk melakukan normalisasi data.

Baris 24 berfungsi untuk mengembalikan *array* ‘fitur’ yang berisi data hasil normalisasi.

5.1.2 Implementasi *K-Means*

5.1.2.1 Inisialisasi Klaster

Sebelum memulai iterasi *K-Means*, langkah pertama yang dilakukan adalah melakukan distribusi data ke setiap klaster secara acak. Berikut merupakan kode sumber untuk menginisialisasi klaster awal setiap data. Implementasi kode sumber untuk mengunggah berkas ditampilkan pada Algoritme 2.

	Algoritme 2: Fungsi Inisialisasi Klaster
1	`public double[][] random_cluster() {`
2	`    Matrix rand = Matrix.random(jum_data, cluster);`
3	`    double[][] init_cl = rand.getArray();`
4	`    for (int i = 0; i < init_cl.length; i++) {`
5	`        double max = Double.MIN_VALUE;`
6	`        for (int j = 0; j < init_cl[0].length; j++) {`
7	`            if (init_cl[i][j] > max) {`
8	`                max = init_cl[i][j];`
9	`            }`
10	`        }`
11	`        for (int j = 0; j < init_cl[0].length; j++) {`
12	`            if (init_cl[i][j] == max) {`
13	`                init_cl[i][j] = 1;`
14	`            } else {`
15	`                init_cl[i][j] = 0;`
16	`            }`
17	`        }`
18	`    }`
19	`    return init_cl;`
20	`}`

Baris 2 berfungsi untuk membangun matriks yang berisi nilai acak dengan jumlah baris sejumlah data dan jumlah kolom sejumlah klaster yang ingin dibentuk.

Baris 6-10 berfungsi untuk mencari nilai maksimal dari setiap baris pada matriks 'init_cl'.

Baris 11-17 berfungsi untuk memberikan nilai 1 pada kolom yang memiliki nilai maksimum setiap baris sesuai dengan pencarian pada baris 6-10, dan memberikan nilai 0 untuk kolom yang lain. Nilai 1 pada setiap baris, menunjukkan bahwa data pada baris tersebut memasuki klaster pada kolom yang bernilai 1.

Baris 19 berfungsi untuk mengembalikan matriks inisialisasi klaster awal dalam *array* 'init_cl'.

5.1.2.2 Menghitung *Centroid*

Untuk menghitung pusat klaster atau *centroid*, memerlukan *array* yang berisi distribusi data ke setiap klaster sebagai parameternya. Keluaran yang dihasilkan adalah nilai rata-rata setiap fitur pada klaster tersebut atau nilai *centroid* setiap klaster. Implementasi kode sumber untuk menghitung pusat klaster ditampilkan pada Algoritme 3.

	Algoritme 3: Fungsi Hitung Centroid
1	`public double[][] hitungCentroid(double[][] cl) {`
2	`    double[] count = new double[cluster];`
3	`    double[][] centroid = new double[cluster][jum_fitur];`
4	`    double[][] temp = new double[cl.length][cluster * jum_fitur];`
5	`    double[] sumCol = new double[temp[0].length];`
6	`    for (int i = 0; i < cl[0].length; i++) {`
7	`        count[i] = 0;`
8	`        for (int j = 0; j < cl.length; j++) {`
9	`            count[i] += cl[j][i];`
10	`        }`
11	`    }`
12	`    int col = 0;`
13	`    for (int i = 0; i < temp.length; i++) {`
14	`        col = 0;`
15	`        for (int j = 0; j < cl[0].length; j++) {`
16	`            for (int k = 0; k < fitur[0].length; k++) {`

```
17                  temp[i][col] = fitur[i][k] * cl[i][j];
18                  col++;
19              }
20          }
21      }
22      for (int i = 0; i < temp[0].length; i++) {
23          sumCol[i] = 0;
24          for (int j = 0; j < temp.length; j++) {
25              sumCol[i] += temp[j][i];
26          }
27      }
28      int clm = 0;
29      for (int k = 0; k < centroid.length; k++) {
30          for (int j = 0; j < centroid[0].length; j++) {
31              centroid[k][j] = sumCol[clm] / count[k];
32              clm++;
33          }
34      }
35      return centroid;
36  }
```

Baris 7-11 berfungsi untuk menghitung banyaknya anggota setiap klaster.

Baris 13-27 berfungsi untuk menghitung jumlah setiap fitur pada setiap klaster.

Baris 29-34 berfungsi untuk menghitung nilai *centroid* dari setiap klaster dengan menghitung nilai rata-rata dari data yang terletak pada klaster yang sama.

Baris 35 berfungsi untuk mengembalikan *array* 'centroid' yang menyimpan nilai *centroid* setiap klaster.

5.1.2.3 Menghitung Jarak Data dengan Pusat Klaster

Setelah mengetahui pusat klaster atau *centroid*, selanjutnya adalah menghitung jarak setiap data dari setiap pusat klaster dengan jarak *Euclidean*. Implementasi kode sumber untuk menghitung jarak data dengan pusat klaster ditampilkan pada Algoritme 4.

```
    Algoritme 4: Fungsi Hitung Jarak
1   public double[][] hitungJarak(double[][] centroid) {
2       double[][] jarak = new double[jum_data][cluster];
3       double temp = 0;
4       for (int i = 0; i < jarak.length; i++) {
5           for (int k = 0; k < jarak[0].length; k++) {
6               temp = 0;
7               for (int j = 0; j < fitur[0].length; j++) {
8                   temp += Math.pow(fitur[i][j] - centroid[k][j], 2);
9               }
10              jarak[i][k] = Math.sqrt(temp);
11          }
12      }
13      return jarak;
14  }
```

Baris 2 berfungsi untuk inisialisasi *array* dua dimensi 'jarak' yang memiliki dimensi baris sebanyak data dan dimensi kolom sebanyak klaster.

Baris 4-12 berfungsi untuk menghitung jarak *Euclidean* antara data dengan pusat klaster. Langkah pertama dengan menjumlahkan nilai kuadrat dari nilai data dikurangi dengan nilai *centroid* klaster tertentu seperti pada baris 7-10 dan selanjutnya menghitung akar kuadrat dari hasil tersebut seperti pada baris 11.

Baris 13 berfungsi untuk mengembalikan *array* 'jarak' yang berisi hasil perhitungan jarak *Euclidean* data dengan *centroid* setiap klaster.

5.1.2.4 Menghitung Objektif

Salah satu kriteria berhenti dalam *K-Means* adalah ketika perubahan nilai objektif lebih kecil atau sama dengan suatu ambang batas atau *threshold*. Sebelum menghitung perubahan nilai objektif, maka perlu dihitung nilai objektif pada iterasi terkini. Implementasi kode sumber untuk menghitung nilai objektif ditampilkan pada Algoritme 5.

	Algoritme 5: Fungsi Hitung Objektif
1	`public double hitungObjektif(double[][] jarak) {`
2	`    clBaru = new double[jarak.length];`
3	`    p = 0;`
4	`    for (int i = 0; i < jarak.length; i++) {`
5	`        double min = Double.MAX_VALUE;`
6	`        for (int k = 0; k < jarak[0].length; k++) {`
7	`            if (jarak[i][k] < min) {`
8	`                min = jarak[i][k];`
9	`                clBaru[i] = k + 1;`
10	`            }`
11	`        }`
12	`        p += min;`
13	`    }`
14	`    return p;`
15	`}`

Baris 2 berfungsi untuk inisialisasi *array* 'clBaru' yang akan menyimpan indeks dari jarak minimum data ke pusat klaster.

Baris 6-11 berfungsi untuk menemukan jarak minimum data dengan pusat klaster dan menyimpan indeks dari jarak minimum tersebut sebagai nilai klaster terbaru dari data.

Baris 12 berfungsi untuk menghitung nilai objektif pada iterasi terkini dengan menjumlahkan jarak minimum semua data dengan pusat klaster.

Baris 14 berfungsi untuk mengembalikan nilai objektif pada iterasi terkini.

5.1.2.5 Menghitung Perubahan Nilai Objektif

Setelah mengetahui nilai objektif pada iterasi terkini, selanjutnya adalah menghitung perubahan nilai objektif iterasi terkini dengan iterasi sebelumnya. Implementasi kode sumber untuk menghitung perubahan nilai objektif ditampilkan pada Algoritme 6.

	Algoritme 6: Fungsi Hitung Delta Objektif
1	`public double deltaP(double pLama) {`
2	`    return Math.abs(p - pLama)`
3	`}`

Baris 3 berfungsi untuk menghitung delta absolut dari perngurangan nilai objektif iterasi terkini dengan iterasi sebelumnya.

5.1.2.6 Memperbarui Klaster

Pada proses menghitung nilai objektif, telah didapatkan klaster terbaru setiap data, pada fungsi ini akan mengubah nomor klaster pada *array* 'clBaru' menjadi

bentuk biner dengan angka 1 sebagai penunjuk letak klaster seperti pada proses inisialisasi klaster. Implementasi kode sumber untuk memperbarui klaster ditampilkan pada Algoritme 7.

	Algoritme 7: Fungsi Update Klaster
1	`public double[][] updateCluster() {`
2	`    double[][] upd_cl = new double[jum_data][cluster];`
3	`    for (int i = 0; i < upd_cl.length; i++) {`
4	`        for (int j = 0; j < upd_cl[0].length; j++) {`
5	`            upd_cl[i][j] = clBaru[i] == j+1 ? 1 : 0;`
6	`        }`
7	`    }`
8	`    return upd_cl;`
9	`}`

Baris 2 berfungsi untuk inisialisasi *array* dua dimensi 'upd_cl' yang akan menyimpan letak klaster terbaru dari setiap data.

Baris 3-7 berfungsi untuk merubah nomor klaster yang sebelumnya disimpan pada *array* 'clBaru' ke dalam bentuk biner. *Array* 'upd_cl' akan berisi nilai 1 pada kolom yang menunjukkan nomor klaster pada setiap data.

Baris 8 berfungsi untuk mengembalikan letak klaster terbaru dari setiap data.

5.1.2.7 Proses *Clustering K-Means*

Proses *clustering* dengan *K-Means* memerlukan iterasi hingga memenuhi kriteria berhenti. Implementasi kode sumber untuk proses *clustering K-means* ditampilkan pada Algoritme 8.

	Algoritme 8: Fungsi Clustering K-Means
1	`public double[][] clustering(int c, int maxIter, double threshold)`
2	`throws IOException{`
3	`    double pInit = 0, pLama = 0, deltaP;`
4	`    double[][] init_cl = random_cluster();`
5	`    double[][] centroid = hitungCentroid(init_cl);`
6	`    double[][] jarak = hitungJarak(centroid);`
7	`    hitungObjektif(jarak);`
8	`    double[][] updt_cl = updateCluster();`
9	`    deltaP(pInit);`
10	`    System.out.println("ITERASI 0 DONE...");`
11	`    int iter = 1;`
12	`    do {`
13	`        double[][] last_cluster = updt_cl;`
14	`        pLama = p;`
15	`        centroid = hitungCentroid(last_cluster);`
16	`        jarak = hitungJarak(centroid);`
17	`        hitungObjektif(jarak);`
18	`        updt_cl = updateCluster();`
19	`        deltaP = deltaP(pLama);`
20	`        System.out.println("ITERASI " + iter + " DONE...");`
21	`        iter++;`
22	`    } while (deltaP > threshold && iter <= maxIter);`
23	`    return centroid;`
24	`}`

Baris 4 berfungsi untuk menginisialisasi *array* 'init_cl' dengan memanggil fungsi 'random_cluster'.

Baris 5 berfungsi untuk menginisialisasi *array* 'centroid' dengan memanggil fungsi 'centroid' dan mengisikan argumen dengan *array* 'init_cl'.

Baris 6 berfungsi untuk menginisialisasi *array* 'jarak' dengan memanggil fungsi 'hitungJarak' dan mengisikan argumen dengan *array* 'centroid'.

Baris 7 berfungsi untuk menghitung nilai objektif iterasi ke-0 dengan memanggil fungsi 'hitungObjektif' dan mengisikan argumen dengan *array* 'jarak'.

Baris 8 berfungsi untuk menginisialisasi *array* 'updt_cl' dengan memanggil fungsi 'updateCluster'.

Baris 9 berfungsi untuk menghitung delta atau perubahan nilai objektif dengan memanggil fungsi deltaP dan mengisikan argumen variabel 'pInit' sebagai nilai objektif awal.

Baris 12-21 berfungsi untuk melakukan iterasi pengelompokan selama kriteria berhenti belum terpenuhi.

Baris 22 berfungsi untuk melakukan pengecekan kriteria perulangan, yaitu ketika nilai deltaP lebih besar dari *threshold* dan iterasi kurang dari sama dengan iterasi maksimum.

Baris 23 berfungsi untuk mengembalikan hasil nilai *centroid* pada iterasi terakhir.

5.1.3 Implementasi RBFNN

5.1.3.1 Menghitung Nilai *Spread*

Untuk mendapatkan nilai *spread* untuk fungsi aktivasi *Gaussian*. Nilai *spread* untuk semua *hidden neuron* pada penelitian ini adalah sama. Implementasi kode sumber untuk menghitung nilai varian setiap klaster yang terbentuk ditampilkan pada Algoritme 9.

	Algoritme 9: Fungsi Hitung Spread
1	`public double[] getSpread(double[][] centroid){`
2	`    double[][] dis = new double[centroid.length][centroid.length];`
3	`    double max_dist = 0;`
4	`    double[] spread = new double[hidneu];`
5	`    for (int i = 0; i < dis.length; i++) {`
6	`        for (int j = 0; j < centroid.length; j++) {`
7	`            double temp = 0;`
8	`            for (int k = 0; k < centroid[0].length; k++) {`
9	`              temp += Math.pow(centroid[i][k] - centroid[j][k], 2);`
10	`            }`
11	`            dis[i][j] = Math.sqrt(temp);`
12	`            if (dis[i][j] > max_dist) {`
13	`                max_dist = dis[i][j];`
14	`            }`
15	`        }`
16	`    }`
17	`    for (int i = 0; i < spread.length; i++) {`
18	`        spread[i] = max_dist / (Math.sqrt(2*centroid.length));`
19	`    }`
20	`    return spread;`
21	`}`

Baris 2 berfungsi untuk inisialisasi *array* 'dis' yang akan menyimpan jarak antar *centroid* klaster.

Baris 4 berfungsi untuk inisialisasi *array* 'spread' yang akan menyimpan nilai *spread* untuk semua *hidden neuron*.

Baris 8-11 berfungsi untuk menghitung jarak *Euclidean* antar *centroid* klaster.

Baris 12-14 berfungsi untuk mencari nilai jarak maksimum antara dua *centroid* klaster.

Baris 17-19 berfungsi untuk menghitung nilai *spread* dan menyimpannya ke dalam *array* 'spread', karena pada penelitian ini nilai *spread* untuk semua *hidden neuron* menggunakan nilai yang sama, maka seluruh indeks *array* 'spread' menyimpan nilai yang sama.

Baris 20 berfungsi untuk mengembalikan nilai *spread*.

5.1.3.2 Menghitung Keluaran *Hidden Layer*

Untuk mendapatkan nilai keluaran dari setiap *hidden neuron*, maka perlu masukan berupa data, *centroid* hasil dari *clustering K-Means* dan nilai *spread*. Keluaran dari *hidden layer* adalah matriks *Gaussian*. Implementasi kode sumber untuk menghitung matriks *Gaussian* sebagai keluaran *hidden layer* ditampilkan pada Algoritme 10.

Algoritme 10: Fungsi Keluaran Hidden Layer

```
1  public double[][] hiddenLayer(double[][] fitur, double[][]
2  centroid, double[] spread) {
3     double[][] gaussian = new double[fitur.length][hidneu];
4     double x_c = 0;
5     for (int k = 0; k < centroid.length; k++) {
6        for (int i = 0; i < fitur.length; i++) {
7           x_c = 0;
8           for (int j = 0; j < fitur[0].length; j++) {
9              x_c += Math.pow(fitur[i][j] - centroid[k][j], 2);
10          }
11          gaussian[i][k] = Math.exp(-(x_c /
12 (2*(Math.pow(spread[k],2)))));
13       }
14    }
15    return gaussian;
16 }
```

Baris 3 berfungsi untuk meginisialisasi *array* dua dimensi 'gaussian' yang akan menyimpan nilai keluaran dari *hidden layer*.

Baris 8-10 berfungsi untuk menghitung jarak antara data dengan *centroid* setiap *hidden neuron*.

Baris 11-12 berfungsi untuk menghitung hasil keluaran *hidden layer* yang kemudian disimpan ke dalam *array* 'gaussian'.

Baris 15 berfungsi untuk mengembalikan hasil keluaran *hidden layer* dalam *array* 'gaussian'.

5.1.3.3 Menghitung Bobot dan Bias

Setelah mendapatkan matriks *Gaussian*, selanjutnya adalah menghitung bobot dan bias dari *hidden layer* ke *output layer*. Sebelum menghitung bobot dan bias, perlu menambahkan 1 kolom terakhir di matriks *Gaussian* untuk nilai bias, yaitu 1. Kemudian baru menghitung bobot dan bias dengan melakukan *pseudoinverse* dari matriks *Gaussian* terbaru. Implementasi kode sumber untuk menghitung bobot dan bias dari *hidden layer* menuju *output layer* ditampilkan pada Algoritme 11.

	Algoritme 11: Fungsi Hitung Bobot dan Bias
1	`public double[][] hitungBobot(double[][] gaussian, double[][]`
2	`target) {`
3	`   double[][] gauss_bias = new`
4	`double[gaussian.length][gaussian[0].length+1];`
5	`   for (int i = 0; i < gaussian.length; i++) {`
6	`       for (int j = 0; j < gaussian[0].length; j++) {`
7	`           gauss_bias[i][j] = gaussian[i][j];`
8	`       }`
9	`       gauss_bias[i][gauss_bias[0].length-1] = 1;`
10	`   }`
11	`   Matrix tar = new Matrix(target);`
12	`   Matrix gauss = new Matrix(gauss_bias); //G`
13	`   Matrix gaussT = gauss.transpose(); //Gt`
14	`   Matrix gtg = gaussT.times(gauss); //Gt*G`
15	`   Matrix g_inv = gtg.inverse(); //inverse`
16	`   Matrix gt = gaussT.times(tar); //Gt*target`
17	`   Matrix bobot = g_inv.times(gt);`
18	`   double[][] weight = bobot.getArray();`
19	`   return weight;`
20	`}`

Baris 5-10 berfungsi untuk menambahkan satu kolom terakhir sebagai bias dengan nilai 1 pada matriks *Gaussian*.

Baris 11 berfungsi untuk menyimpan matriks target.

Baris 12 berfungsi untuk menyimpan matriks *Gaussian* terbaru ke dalam Matrix JAMA.

Baris 13 berfungsi untuk melakukan *transpose* terhadap matriks *Gaussian*.

Baris 14 berfungsi untuk melakukan perkalian matriks hasil *transpose* pada baris 13 dengan matriks *Gaussian*.

Baris 15 berfungsi untuk melakukan *inverse* terhadap operasi matriks pada baris 14.

Baris 16 berfungsi untuk melakukan perkalian matriks antara matriks hasil *transpose Gaussian* dengan matriks target.

Baris 17 berfungsi untuk menyimpan nilai bobot dari hasil *pseudoinverse* matriks *Gaussian* sehingga menghasilkan bobot.

Baris 19 berfungsi untuk mengembalikan hasil matriks bobot dalam *array* 'weight'.

5.1.3.4 Menghitung Keluaran *Output Layer*

Untuk mengetahui hasil dari klasifikasi, maka perlu dilakukan langkah terakhir yaitu menghitung keluaran dari setiap *output neuron*. Implementasi kode sumber untuk menghitung keluaran *output layer* ditampilkan pada Algoritme 12.

	Algoritme 12: Fungsi Hitung Keluaran Output Layer
1	`public String[] outputLayer(double[][] bobot, double[][] gaussian){`
2	`  double[][] outp = new double[gaussian.length][gaussian[0].length];`
3	`    String[][] label =`
4	`{{"0","CL0"},{"1","CL1"},{"2","CL2"},{"3","CL3"},`
5	`{"4","CL4"},{"5","CL5"},{"6","CL6"}};`
6	`   double w_g = 0;`
7	`   double[][] w = new double[bobot.length-1][bobot[0].length];`
8	`   double[] b = new double[bobot[0].length];`
9	`   //Pisahkan w dan b`

```
10      for (int i = 0; i < bobot.length-1; i++) {
11          for (int j = 0; j < bobot[0].length; j++) {
12              w[i][j] = bobot[i][j];
13          }
14      }
15      for (int i = 0; i < bobot[0].length; i++) {
16          b[i] = bobot[bobot.length-1][i];
17      }
18      //output
19      Matrix weight = new Matrix(w);
20      Matrix g = new Matrix(gaussian);
21      Matrix out = g.times(weight);
22      outp = out.getArray();
23      for (int i = 0; i < outp.length; i++) {
24          for (int j = 0; j < outp[0].length; j++) {
25              outp[i][j] += b[j];
26          }
27      }
28      //Menentukan kelas (argmax output)
29      double[] klas = new double[outp.length];
30      for (int i = 0; i < outp.length; i++) {
31          double max = Double.MIN_VALUE;
32          for (int j = 0; j < outp[0].length; j++) {
33              if (outp[i][j] > max) {
34                  max = outp[i][j];
35                  klas[i] = j;
36              }
37          }
38      }
39      System.out.println("============================");
40      System.out.println("===== HASIL KLASIFIKASI =====");
41      System.out.println("============================");
42      System.out.println("\tDATA\tKELAS");
43      String[] hasil = new String[klas.length];
44      for (int i = 0; i < klas.length; i++) {
45          for (int j = 0; j < label.length; j++) {
46              double temp = Double.valueOf(label[j][0]);
47              if (klas[i] == temp) {
48                  hasil[i] = label[j][1];
49              }
50          }
51          System.out.println("\t"+(i+1)+"\t"+hasil[i]);
52      }
53      return hasil;
54  }
```

Baris 10-14 berfungsi untuk menyimpan nilai bobot dari *hidden layer* menuju *output layer* dari hasil yang didapatkan dari hasil *pseudoinverse* matriks *Gaussian* ke dalam *array* 'w'.

Baris 15-17 berfungsi untuk menyimpan bobot bias ke semua *output neuron*, yaitu baris terakhir pada *array* bobot yang didapat dari hasil *pseudoinverse* matriks *Gaussian* ke dalam *array* 'b'.

Baris 19-22 berfungsi untuk menghitung keluaran dari *output layer* dengan mengalikan matriks *Gaussian* dengan matriks bobot.

Baris 23-27 berfungsi untuk menambahkan hasil keluaran *output layer* dengan matriks bias sehingga mendapatkan hasil akhir keluaran *output layer*.

Baris 30-38 berfungsi untuk menemukan nilai keluaran terbesar dari output *neuron* untuk setiap data dan menentukan hasil klasifikasi untuk data tersebut.

Baris 44-50 berfungsi untuk memberikan label kelas untuk setiap data.

Baris 53 berfungsi untuk mengembalikan hasil klasifikasi data dalam *array* 'hasil'.

5.1.3.5 Menghitung Akurasi

Setelah mendapatkan hasil keluaran dari *output layer*, proses berikutnya adalah mengetahui akurasi dari hasil klasifikasi. Proses ini menghitung banyaknya kelas hasil prediksi yang sesuai dengan kelas aktual data. Implementasi kode sumber untuk menghitung akurasi ditampilkan pada Algoritme 13.

Algoritme 13: Fungsi Menghitung Akurasi

```
1  public void akurasi(String[] hasil, String[] kelas){
2     int count = 0;
3     for (int i = 0; i < hasil.length; i++) {
4         if (hasil[i].equalsIgnoreCase(kelas[i])) {
5             count++;
6         }
7     }
8     System.out.println("\n============================");
9     System.out.println("======= HASIL AKURASI =======");
10    System.out.println("============================");
11    double acc = ((double)count/hasil.length)*100;
12    System.out.println(" Correctly Classified = "+count);
13    System.out.println(" Incorrectly Classified = "+(hasil.length-
14 count));
15     System.out.println(" Accuracy = "+String.format("%.2f",acc)+"%");
16 }
```

Baris 3-7 berfungsi untuk menghitung banyaknya data yang diklasifikasikan dengan benar.

Baris 11 berfungsi untuk menghitung nilai akurasi hasil klasifikasi.

Baris 12 berfungsi untuk mencetak banyaknya data yang diklasifikasikan dengan benar.

Baris 13-14 berfungsi untuk mencetak banyaknya data yang diklasifikasikan dengan tidak benar.

Baris 15 berfungsi untuk mencetak hasil akurasi.

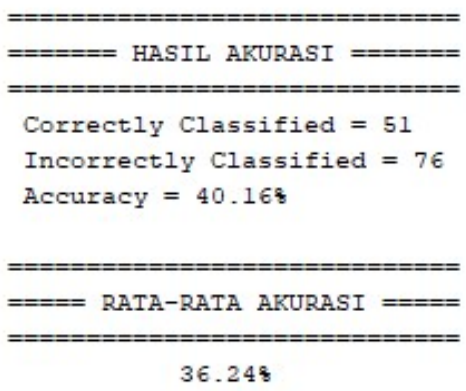
5.2 Antarmuka

Untuk mengetahui hasil klasifikasi dari setiap data, maka ditampilkan hasil seperti pada Gambar 5.1.

```
============================
===== HASIL KLASIFIKASI =====
============================
        DATA    KELAS
        1       CL2
        2       CL0
        3       CL5
        4       CL0
        5       CL1
```

Gambar 5.1 Antarmuka Hasil Klasifikasi

Kemudian untuk mengetahui akurasi dari sistem klasifikasi maka ditampilkan rincian hasil akurasi sistem dan rata-rata akurasi dari 5 *fold* data seperti pada Gambar 5.2.



Gambar 5.2 Antarmuka Hasil Akurasi

BAB 6 PENGUJIAN DAN ANALISIS

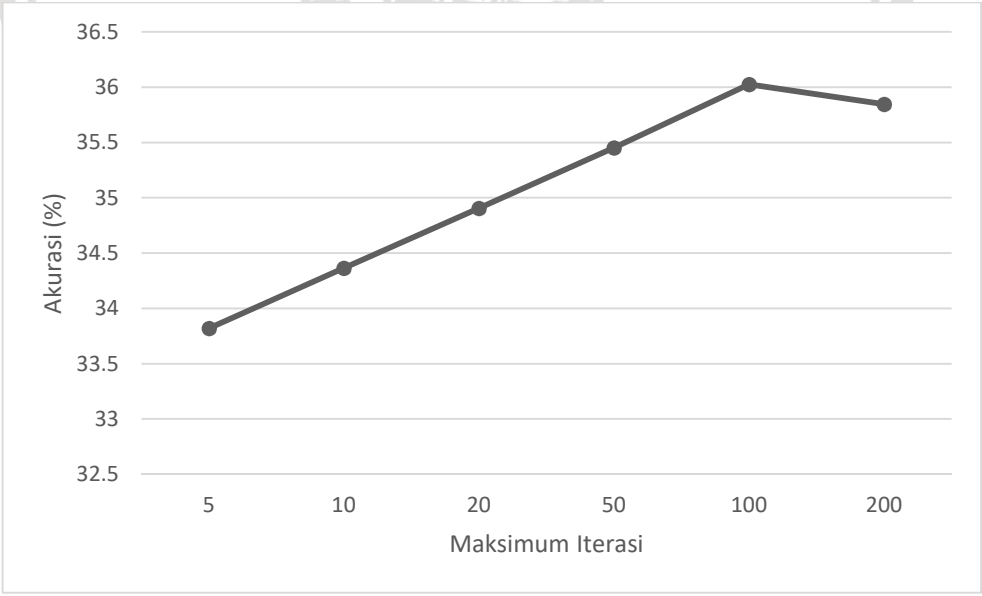
6.1 Hasil Pengujian dan Analisis Maksimum Iterasi *K-Means*

Pada metode *K-Means*, salah satu kriteria berhenti adalah ketika jumlah iterasi telah mencapai batas maksimum iterasi. Pengujian batas maksimum iterasi *K-Means* bertujuan untuk mengetahui pengaruh batas maksimum iterasi dengan akurasi sistem klasifikasi. Akurasi dari hasil klasifikasi, didapatkan dengan menggunakan metode *K-Fold Cross Validation* dengan 5 *fold*. Karena pada *K-Means* inisialisasi klaster dilakukan secara acak, maka setiap pengujian dilakukan sebanyak 5 percobaan. Jumlah maksimum iterasi yang diujikan adalah 5, 10, 20, 50, 100 dan 200. Jumlah *hidden neuron* yang digunakan pada pengujian ini adalah 10. Hasil dari pengujian maksimum iterasi *K-Means* ditampilkan pada Tabel 6.1.

Tabel 6.1 Hasil Pengujian Maksimum Iterasi K-Means

Maksimum Iterasi	Percobaan					Rata-Rata Akurasi
	1	2	3	4	5	
5	33,377%	34,011%	35,13%	33,526%	33,058%	33,82%
10	35,135%	33,854%	34,813%	34,18%	34,336%	34,464%
20	34,492%	34,493%	35,607%	34,652%	35,281%	34,906%
50	35,456%	35,137%	35,453%	35,613%	35,608%	35,453%
100	37,052%	35,936%	35,446%	35,448%	36,251%	**36,027%**
200	35,936%	35,621%	36,661%	35,929%	36,087%	35,847%

Grafik rata-rata akurasi dari hasil pengujian maksimum iterasi pada Tabel 6.1 ditampilkan pada Gambar 6.1.



Gambar 6.1 Grafik Pengujian Maksimum Iterasi *K-Means*

Dari hasil pengujian pada Gambar 6.1, rata-rata akurasi dari lima percobaan setiap jumlah iterasi pengujian menunjukkan bahwa semakin banyak jumlah maksimum iterasi, maka nilai akurasi juga semakin meningkat hingga batas maksimum iterasi bernilai 100, setelah itu mengalami penurunan 0,18% ketika batas maksimum iterasi bernilai 200. Pada pengujian batas maksimum iterasi bernilai 5, rata-rata akurasi yang dihasilkan memiliki nilai terendah yaitu 33,82%. Sedangkan rata-rata akurasi tertinggi terjadi ketika maksimum iterasi bernilai 100 yaitu 36,027%. Hal ini dapat terjadi karena, apabila iterasi *clustering K*-Means dihentikan ketika masih banyak data yang berada pada klaster yang belum tepat, maka kualitas klaster yang terbentuk juga semakin rendah. Kualitas klaster yang rendah akan berpengaruh terhadap *centroid* yang terbentuk. Sehingga apabila *centroid* tidak membentuk pola data rata-rata kelas dengan baik, maka hasil akurasi klasifikasi juga akan rendah.

6.2 Hasil Pengujian dan Analisis Banyak *Hidden Neuron*

Pada RBFNN, banyak *hidden neuron* merepresentasikan banyak klaster yang dibentuk dari proses *K-Means*. Pengujian banyak *hidden neuron* bertujuan untuk mengetahui banyak *hidden neuron* terbaik untuk hasil klasifikasi dan mengetahui pengaruh banyak *hidden neuron* dengan akurasi sistem klasifikasi. Karena pada metode RBFNN terdapat proses *clustering* dengan *K-Means* yang mana inisialisasi klaster dilakukan secara acak, maka setiap program dijalankan akan memberikan hasil yang bebeda. Sehingga pengujian banyak *hidden neuron* dilakukan percobaan sebanyak 5 percobaan. Banyak *hidden neuron* yang diujikan adalah 2, 3, 4, 5, 6, 7, 8, 9, 10, 20 dan 50. Parameter lain yaitu batas iterasi maksimum *K-Means* yang digunakan pada pengujian ini adalah 100 dan evaluasi hasil akurasi menggunakan metode *K-Fold* sebanyak 5 *fold*. Hasil dari pengujian banyak *hidden neuron* ditampilkan pada Tabel 6.2.

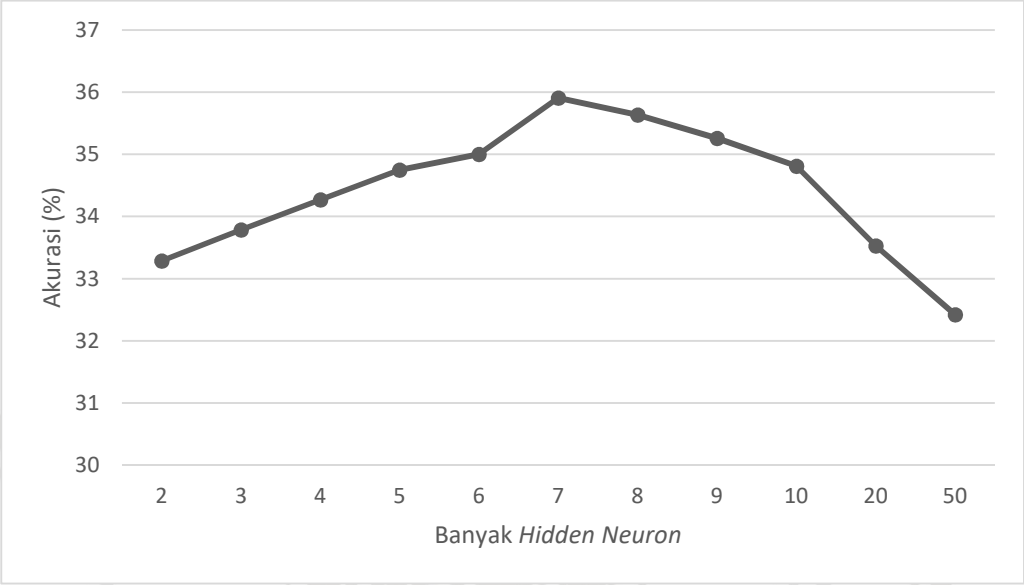
Tabel 6.2 Hasil Pengujian Banyak *Hidden Neuron*

Hidden Neuron	Percobaan					Rata-Rata Akurasi
	1	2	3	4	5	
2	33,067%	33,062%	33,382%	33,707%	33,222%	33,288%
3	33,846%	33,528%	33,849%	34,162%	33,541%	33,785%
4	34,018%	34,332%	34,498%	33,853%	34,642%	34,269%
5	34,648%	34,175%	34,494%	35,3%	35,127%	34,749%
6	34,969%	34,808%	35,123%	34,968%	35,137%	35,001%
7	35,769%	36,296%	36,256%	35,612%	35,607%	**35,908%**
8	36,084%	35,446%	35,283%	35,759%	35,613%	35,637%
9	34,957%	35,773%	35,135%	35,298%	35,123%	35,257%
10	34,817%	34,957%	34,486%	34,501%	35,292%	34,811%
20	33,683%	33,703%	33,214%	33,211%	33,838%	33,53%

Tabel 6.2 Hasil Pengujian Banyak *Hidden Neuron* (lanjutan)

Hidden Neuron	1	2	3	4	5	Rata-Rata Akurasi
50	32,423%	32,423%	32,423%	32,423%	32,423%	32,423%

Grafik rata-rata akurasi dari hasil pengujian banyak *hidden neuron* pada Tabel 6.2 ditampilkan pada Gambar 6.2.



Gambar 6.2 Grafik Pengujian Banyak *Hidden Neuron*

Dari hasil pengujian pada Gambar 6.2, rata-rata akurasi dari lima percobaan setiap jumlah *hidden neuron* menunjukkan bahwa jumlah *hidden neuron* yang memiliki hasil tertinggi yaitu 7 *hidden neuron* dengan rata-rata akurasi sebesar 35,908%. Dari grafik menunjukkan bahwa akurasi akan naik hingga mencapai puncaknya yaitu ketika *hidden neuron* berjumlah 7, setelah itu terus menurun hingga yang terendah ketika jumlah *hidden neuron* 50 dengan akurasi 32,423%. Hal ini terjadi karena dengan jumlah *hidden neuron* 7, menunjukkan bahwa dengan 7 klaster yang terbentuk maka kualitas klaster paling optimal dibandingkan dengan jumlah klaster kurang atau lebih dari 7. Sehingga nilai rata-rata setiap klaster atau *centroid* klaster merepresentasikan nilai yang sesuai dengan rata-rata dari 7 kelas data latih dengan baik. Apabila banyaknya *hidden neuron* terlalu sedikit, maka akan terjadi *underfitting*, yaitu keadaan dimana terlalu sedikit pola data yang dikenali. Sedangkan apabila banyaknya *hidden neuron* terlalu banyak, maka akan terjadi *overfitting*, yaitu keadaan dimana pola data yang dikenali terlalu banyak, sehingga ketika dilakukan pengujian dengan data uji yang belum pernah dikenali polanya, akan diklasifikasian dengan tidak benar.

6.3 Analisis Global

Pada penelitian ini, terdapat dua jenis pengujian yang dilakukan terkait dengan metode RBFNN. Yang pertama adalah pengujian mengenai pengaruh batas maksimum iterasi pada *K-Means* terhadap akurasi klasifikasi RBFNN. Pada pengujian tersebut menunjukkan bahwa akurasi klasifikasi terus meningkat seiring dengan bertambahnya batas iterasi maksimum untuk metode *K-Means* hingga mencapai puncaknya yaitu dengan batas 100 iterasi. Kemudian dengan melakukan pengujian batas maksimum iterasi di atas 100, yaitu 200, ternyata hasil akurasi menurun sebesar 0,18%. Kemudian pengujian kedua mengenai pengaruh jumlah *hidden neuron* pada jaringan RBF terhadap akurasi klasifikasi. Pada pengujian tersebut menunjukkan bahwa jumlah *hidden neuron* sebanyak 7 menjadi jumlah yang paling optimal. Sedangkan jika *hidden neuron* sejumlah 50, maka akurasi yang dicapai jauh menurun sebesar 3,483%.

Dari dua pengujian yang telah dilakukan, hasil akurasi rata-rata jika parameter yang digunakan adalah parameter optimal dari dua pengujian, yaitu batas iterasi maksimum sebanyak 100 dan *hidden neuron* sebanyak 7, hanya sebesar 35,908%. Hasil yang didapat ini masih rendah untuk akurasi suatu metode klasifikasi. Faktor yang dapat menyebabkan rendahnya akurasi klasifikasi pada penelitian ini adalah nilai rata-rata dari fitur oscore, ascore dan cscore pada beberapa kelas cenderung mirip.

Pada penelitian ini data yang digunakan memiliki 7 kelas untuk klasifikasi. Dari ketujuh kelas tersebut, terdapat beberapa kelas yang memiliki karakteristik data yang hampir mirip, terutama pada nilai fitur Oscore, Ascore dan Cscore. Hal ini dapat dibuktikan dengan menghitung nilai rata-rata fitur Oscore, Ascore dan Cscore pada setiap kelas. Hasil rata-rata fitur tersebut pada setiap kelas ditampilkan pada Tabel 6.3.

Tabel 6.3 Rata-Rata Fitur Oscore, Ascore dan Cscore Pada Setiap Kelas

Kelas	Fitur		
	Oscore	Ascore	Cscore
CL0	41,7	44,45	45,23
CL1	43,4	43,92	43,37
CL2	43,97	42,34	41,75
CL3	45,76	42,5	41,56
CL4	46,04	40,88	40,25
CL5	45,45	42,24	38,79
CL6	47,43	42,11	40,64

Dari Tabel 6.3, pada kolom untuk fitur Oscore, kelas CL1 dan CL2 memiliki nilai rata-rata Oscore yang hampir sama, keduanya memiliki selisih nilai hanya sebesar 0,57. Kemudian untuk kelas CL3, CL4 dan CL5 memiliki nilai rata-rata Oscore yang hampir sama juga, ketiganya memiliki selisih nilai yang tidak lebih dari 0,6.

Sedangkan untuk kelas CL0 dan CL6 memiliki selisih nilai yang jauh berbeda yaitu sebesar 5,73. Begitu pula selisih nilai antara kelas CL0 atau CL6 dengan kelas yang lainnya yang memiliki selisih nilai lebih dari 1,3. Selanjutnya pada kolom untuk fitur Ascore, kelas CL0 dan CL1 memiliki nilai rata-rata Ascore yang hampir sama, keduanya memiliki selisih nilai hanya sebesar 0,53. Kemudian untuk kelas CL2, CL3, CL5 dan CL6 juga memiliki nilai rata-rata Ascore yang hampir sama, keempatnya memiliki selisih nilai tidak lebih dari 0,4. Selanjutnya pada kolom Cscore, kelas CL2 dan CL3 memiliki nilai rata-rata Cscore yang hampir sama, yaitu hanya memiliki selsisih sebesar 0,19. Kemudian untuk CL4 dan CL6 memiliki selisih nilai hanya sebesar 0,39.

Selanjutnya, dilakukan analisis terhadap rata-rata fitur usia, pendidikan dan *impulsivity* dari setiap kelas, untuk mengetahui keterkaitan ketiga fitur ini dengan perbedaan karakteristik antar kelas. Hasil rata-rata fitur usia, pendidikan dan *impulsivity* setiap kelas ditampilkan pada Tabel 6.4.

Tabel 6.4 Rata-Rata Fitur Usia, Pendidikan dan *Impulsivity* Setiap Kelas

Kelas	Fitur		
	Usia	Pendidikan	Impulsivity
CL0	3,315	6,094	4,468
CL1	3,522	6,391	5,365
CL2	2,754	6,246	5,861
CL3	2,519	6,13	6,759
CL4	2,375	5,958	7,167
CL5	2,414	5,276	6,759
CL6	2,563	5,25	7,35

Dari hasil rata-rata fitur pada Tabel 6.4, dapat ditunjukkan bahwa kelas CL0 dan CL1 memiliki nilai yang dekat untuk rata-rata fitur usia yaitu dengan perbedaan 0,207, sedangkan untuk kelas lainnya berada pada rentang yang sama yaitu memiliki rata-rata di bawah 3. Kemudian untuk fitur pendidikan, kelas CL0, Cl1, CL2 dan CL3 memiliki kemiripan rata-rata nilai fitur dengan bernilai lebih dari 6. Sedangkan kelas sisanya memiliki rata-rata fitur bernilai kurang dari 6. Untuk fitur *impulsivity* kelas CL0 dan CL1 memiliki nilai rata-rata kurang dari 5,5, sedangkan untuk kelas sisanya memiliki nilai rata-rata di atas 5,5. Dari hasil Tabel 6.4 dapat ditunjukkan bahwa kelas CL0 dan CL1 memiliki kemiripan terhadap ketiga fitur tersebut begitupula dengan kelas CL2, CL3, CL4, CL5 dan CL6 yang memiliki kemiripan. Dari hasi analisis tersebut, dapat disimpulkan bahwa jumlah kelas yang banyak dan rata-rata nilai atribut setiap kelas hampir sama atau tidak unik, menyebabkan pembentukan model klasifikasi menjadi kurang baik. Sehingga ketika dilakukan pengujian, maka banyak data yang dikelompokkan ke kelas yang tidak sesuai dengan kelas aktual melainkan ke kelas yang memiliki nilai atribut hampir mirip dengan kelas aktualnya. Terlebih lagi, jumlah data setiap kelas tidak merata. Sehingga kelas yang memiliki jumlah data lebih sedikit tidak

terklasifikasikan dengan baik. Sebaliknya, kelas yang memiliki jumlah data lebih banyak akan mendominasi hasil klasifikasi.

Berikutnya, dilakukan pengujian dengan mengklasifikasikan data menjadi dua kelas, yaitu CL0 dan CL1 menjadi kelas *Non-user* dan sisanya menjadi kelas *User*. Karena menurut penelitian yang dilakukan oleh Fehrman (2015), pengguna narkoba yang tidak mengonsumsi narkoba dalam sepuluh tahun terakhir dapat dikategorikan sebagai bukan pemakai atau *non*-user, sedangkan yang masih mengonsumsi narkoba dalam kurun waktu sepuluh tahun terakhir dikategorikan sebagai pemakai atau *user*. Selain itu, terdapat kemiripan terhadap rata-rata nilai fitur untuk kelas CL0 dan CL1 serta CL2, CL3, CL4, CL5 dan CL6 seperti ditampilkan pada Tabel 6.3 dan Tabel 6.4. Pengujian ini dilakukan untuk membuktikan pengaruh jumlah kelas dan kemiripan karakteristik tiap kelas yang terlalu dekat dapat memengaruhi hasil akurasi. Hasil pengujian dengan menggunakan dua kelas, *hidden neuron* sebanyak 7 dan batas maksimum iterasi *K-Means* sebanyak 100 ditampilkan pada Tabel 6.5.

Tabel 6.5 Hasil Pengujian 2 Kelas

Percobaan	Akurasi
1	70,88%
2	70,73%
3	71,63%
4	72,36%
5	70,99%
Rata-rata	**71,32%**

Dari hasil percobaan menggunakan dua kelas pada Tabel 6.5, menunjukkan bahwa rata-rata akurasi yang didapatkan jauh lebih baik dibandingkan dengan menggunakan 7 kelas yaitu dengan rata-rata akurasi sebesar 71,32%. Hal ini mengindikasikan bahwa ketika kriteria data antar kelas memiliki nilai yang jauh berbeda, maka proses *training* menjadi lebih efektif. Proses *training* yang efektif akan menghasilkan pola klasifikasi yang baik sehingga data dapat diklasifikasikan dengan baik pula.

BAB 7 PENUTUP

7.1 Kesimpulan

Dari hasil penelitian yang telah dilakukan bertahap mulai dari proses perancangan, implementasi, serta pengujian dan analisis mengenai Penentuan Waktu Terakhir Penggunaan Ganja dengan Metode *Radial Basis Function Neural Network* (RBFNN) dapat disimpulkan beberapa hal, diantaranya:

1. Setelah melakukan pengujian terhadap metode RBFNN dengan *K-Means*, didapatkan parameter optimal yang menghasilkan akurasi paling tinggi. Parameter tersebut diantaranya adalah jumlah *hidden neuron* sebanyak 7 *neuron* dan batas iterasi maksimum untuk metode *K-Means* adalah sejumlah 100.
2. Dengan menggunakan dua parameter optimal yaitu 7 *hidden neuron* dan batas iterasi maksimum *K-Means* sebanyak 100, hasil akurasi yang dihasilkan adalah sebesar 35,908%. Hasil akurasi yang rendah ini disebabkan karena jumlah kelas yang banyak tetapi karakteristik data setiap kelas yang hampir sama dan jumlah data setiap kelas yang tidak merata, sehingga proses *training* menjadi tidak efektif. Akibatnya ketika dilakukan pengujian menggunakan data uji, banyak data yang diklasifikasikan cenderung ke kelas yang memiliki data paling banyak atau ke kelas yang memiliki karakteristik yang hampir sama dengan data uji tersebut. Penggabungan kelas yang memiliki karakteristik data yang sama yaitu kelas CL0 dan CL1 menjadi kelompok *non-user* dan CL2, CL3, CL4, CL5, CL6 menjadi kelompok *user*, terbukti mampu memberikan hasil akurasi yang jauh lebih baik, yakni sebesar 71,32%.

7.2 Saran

Setelah mengetahui hasil dari penelitian ini, diharapkan penelitian selanjutnya memberikan hasil yang lebih baik. Berikut beberapa saran yang dapat digunakan untuk penelitian selanjutnya.

1. Menggunakan banyak data yang seimbang untuk setiap kelas, sehingga tidak ada kecenderungan klasifikasi terhadap kelas yang memiliki banyak data lebih banyak.
2. Menggunakan fitur yang dapat membedakan antar kelas, sehingga dapat membentuk pola klasifikasi dengan baik.
3. Menggunakan metode seleksi fitur untuk menentukan fitur yang optimal sehingga hasil klasifikasi lebih baik.

DAFTAR PUSTAKA

Antara News, 2015. *Jenis Rehabilitasi Harus Disesuaikan dengan Tingkat Penggunaan Narkoba*. [online] Antara News. Tersedia di: <https://www.antaranews.com/berita/473092/jenis-rehabilitasi-harus-disesuaikan-dengan-tingkat-penggunaan-narkoba> [Diakses 16 Agustus 2018]

Astuti, N., Soesanto, O., Kartini, D., 2017. Algoritma Fuzzy C-Means (FCM) untuk Penentuan Nilai Center Radial Basis Function (RBF) pada Klasifikasi Data Penyakit Karies Gigi. *Jurnal Elektronik Nasional Teknologi dan Ilmu Komputer*, p.92-101.

Azmi, Fadhillah, 2016. Analisis Learning Jaringan RBF (Radial Basis Function Network) pada Pengenalan Pola Alfanumerik. *Jurnal TIMES*, 5(2), p.32-34.

Badan Narkotika Nasional Baddoka Makassar, 2017. *Rehabilitasi*. [online] Badan Narkotika Nasional Baddoka Makassar. Tersedia di: < http://rehabbaddoka.com/ > [Diakses 30 September 2018]

Badan Narkotika Nasional Republik Indonesia, 2017. *Data Pendukung Press Release Akhir Tahun 2017*. [pdf] Badan Narkotika Nasional Republik Indonesia. Tersedia di: < http://www.bnn.go.id/_multimedia/document/20180208/lampiran_press_release_akhir_tahun_2017_fin-20180208110343.pdf> [Diakses 18 Juli 2018]

Chamidah, N., Wiharto, Salamah, U., 2012. Pengaruh Normalisasi Data pada Jaringan Syaraf Tiruan Backpropagasi Gradient Descent Adaptive Gain (BPGDAG) untuk Klasifikasi. *Jurnal ITSMAR*, 1(1), p.28-33.

Dillak, R. Y., Bintiri, M. G., Sina, D. R., 2012. *Penerapan Jaringan Saraf Tiruan Radial Basis Function pada Diaknosa dan Medical Presription Penyakit Jantung*. Dalam: Seminar Nasional Informatika, 2012. Yogyakarta, 30 Juni 2012.

Fehrman, E., Muhammad, A.K., Mirkes, E.M., Egan, V. & Gorban, A.N., 2017. *Drug Consumption (Quantified) Dataset*. UCI Machine Learning [online] Tersedia melalui: UCI Machine Learning Repository < https://archive.ics.uci.edu/ml/datasets/Drug+consumption+(quantified)> [Diakses 12 Juli 2018]

Fehrman, E., Muhammad, A.K., Mirkes, E.M., Egan, V. & Gorban, A.N., 2017. *The Five Factor Model of Personality and Evaluation of Drug Comsumption Risk*. Cornell University [online] Tersedia melalui: Cornell University Library <https://arxiv.org/abs/1506.06297> [Diakses 12 Juli 2018]

Feist, J., Feist, G.J., 2009. *Theories of Personality*. New York: Plenum Press.

Friedman, H.S., Schustack, M.W., 2012. *Personality: Classis Theories and Modern Research*. Boston: Pearson.

Hello Sehat, 2017. *Tanda Gejala Orang Sakau Ganja*. [online] Hello Sehat. Tersedia di: <https://hellosehat.com/hidup-sehat/tips-sehat/tanda-gejala-orang-sakau-ganja/> [Diakses 15 Agustus 2018]

Humas Badan Narkotika Nasional, 2011. *Ulasan Tentang Ganja*. [online] DEDI Humas BNN. Tersedia di: < http://dedihumas.bnn.go.id/read/section/artikel/2012/04/02/354/ulasan-tentang-ganja> [Diakses 15 Agustus 2018]

Kompas, 2017. *Buwas: Indonesia Darurat Narkoba Sejak 1971 Sampai Sekarang*. [online] Kompas. Tersedia di: < https://regional.kompas.com/read/2017/11/02/17045461/buwas-indonesia-darurat-narkoba-sejak-1971-sampai-sekarang> [Diakses 18 Juli 2018]

Kuntjoro, Dwi A., Setiawan, Budi D., Perdana, Rizal S., 2018. *Algoritme Genetika untuk Optimasi K-Means Clustering dalam Pengelompokan Data Tsunami*. S1. Universitas Brawijaya.

Mirawanti, Y., Ulama, Brodjol S. S., 2012. *Perbandingan Metode Regresi Logistik Ordinal dengan Jaringan Saraf Tiruan Fungsi Radial Basis Studi Kasus Klasifikasi Rumahtangga Miskin Kota Pasuruan Tahun 2008*. S2. Institut Teknologi Sepuluh Nopember.

Partodiharjo, Subagyo, 2006. *Kenali Narkoba dan Musuhi Penyalahgunaannya*. Jakarta: Erlangga.

Patton, J.H., Stanford, M.S., Barrat, E.S., 1995. Factor Structure of The Barrat Impulsiveness Scale. *Journal of Clinical Psychology*, 51, p.764-768.

Pratiwi, M., Alexander, Harefa, J., Nanda, S., 2015. *Mammograms Classification using Gray-level Co-occurrence Matrix and radial Basis Function Neural Network*. Dalam: International Conference on Computer Science and Computational Intelligence, 2015, p.83-91.

Pusat Penelitian Data dan Informasi Badan Narkotika Nasional Republik Indonesia, 2017. *Survei Nasional Penyalahgunaan Narkoba di 34 Provinsi Tahun 2017*. [pdf] Pusat Penelitian Data dan Informasi Badan Narkotika Nasional Republik Indonesia. Tersedia di: < http://www.bnn.go.id/_multimedia/document/20180508/BUKU_HASIL_LIT_2017.pdf > [Diakses 14 Agustus 2018]

Rafaeilzadeh, P., Tang, L., Liu, H., 2009. Cross-Validation. *Encyclopedia of Database Systems*, p.532-538.

Santosa, S., Widjanarko, A., Supriyanto, C., 2016. Model Prediksi Ginjal Kronik Menggunakan Radial Basis Function. *Jurnal Pseudocode*, 3(2), p.163-170.

Undang-undang Republik Indonesia nomor 35 tahun 2009 tentang Narkotika. Jakarta: Presiden Republik Indonesia.

Wutsqa, D. U., Mandadara, H. L. R., 2017. *Lung Cancer Classification using Radial Basis Function Neural Network Model with Point Operation*. Dalam: 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), 2017.

Yayasan Pelita Ilmu, 2017. *Mengenali Gejala Orang yang Sakau Narkoba Jenis Sabu*. [online] Yayasan Pelita Ilmu. Tersedia di: <ypi.or.id/656-2> [Diakses 15 Agustus 2018]

Yayasan Pelita Ilmu, 2017. *Tahapan Rehabilitasi Narkoba*. [online] Yayasan Pelita Ilmu. Tersedia di: < http://ypi.or.id/tahapan-rehabilitasi-narkoba/> [Diakses 28 Agustus 2018]